Санкт-Петербургский государственный институт точной механики и оптики
(технический университет)

Кафедра “Компьютерные технологии”

Корниенко А.А., Курочкин Ю.В., Шалыто А.А.

Эмуляция пользовательского оконного интерфейса

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

Содержание

Я рисую,
Я тебя рисую,
Стоя у окна.
Из песни

Введение

В настоящее время наиболее часто применяемым пользовательским интерфейсом является оконный. Существует множество различных вариантов реализации оконного интерфейса для разнообразных платформ и целей, использующих разные средства и технологии программирования.

В данной работе предлагается применить SWITCH-технологию для реализации логики интерфейса указанного типа. Эта технология предложена А.А. Шалыто и развита им совместно с Н.И. Туккелем. Ознакомиться с этой технологией и с примерами ее использования можно на сайтах http://is.ifmo.ru и http://www.softcraft.ru.

В работе применяется объектный подход с явным выделением состояний. Программа является событийной. Она написана на языке C++ (Visual Studio .NET).

1. Постановка задач

Цель работы состоит в создании программы, эмулирующей простейший оконный пользовательский интерфейс. Программа ***эмулирует собственный рабочий стол***, на котором могут быть открыты ***окна***.

Каждое окно (рис. 1) имеет заголовок (серый прямоугольник) и клиентскую часть (синий квадрат). За заголовок окно можно перетаскивать по экрану. На заголовке есть кнопки минимизации (слева) и закрытия (справа) окна. В правом нижнем углу клиентской части расположена зона изменения размера окна, при перетаскивании за которую, окно изменяет размер.

Рис. 1. Пример окна

На рабочем столе программы можно создать большое количество окон (рис. 2). При этом в каждый момент времени активным может быть не более одного окна. Активное окно отображается поверх остальных окон и имеет по сравнению с ними более темный цвет заголовка.

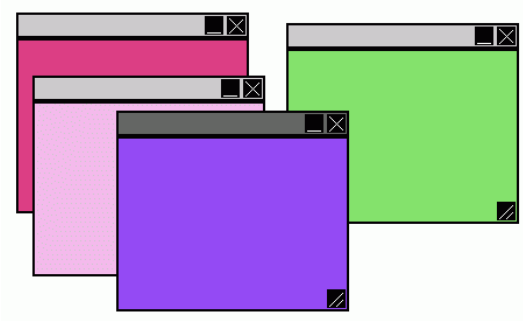

Рис. 2. Пример нескольких окон

2. Диаграмма классов

Программа состоит из трех основных классов: “Window”, “WindowManager” и “Dispatcher”. Кроме того, она содержит стандартные классы среды разработки, сгенерированные автоматически. Они не показаны на диаграмме классов.

На рис. 3 приведена диаграмма основных классов, выполненная в нотации языка UML. Каждый класс является автоматным – содержит автомат, определяющий его поведение. Входные переменные *bool xi()*, автоматная функция *void Aj(int e)* и функции выходных воздействий *void zj_k()* являются методами этого класса. Кроме того, каждый класс содержит конструктор и деструктор.

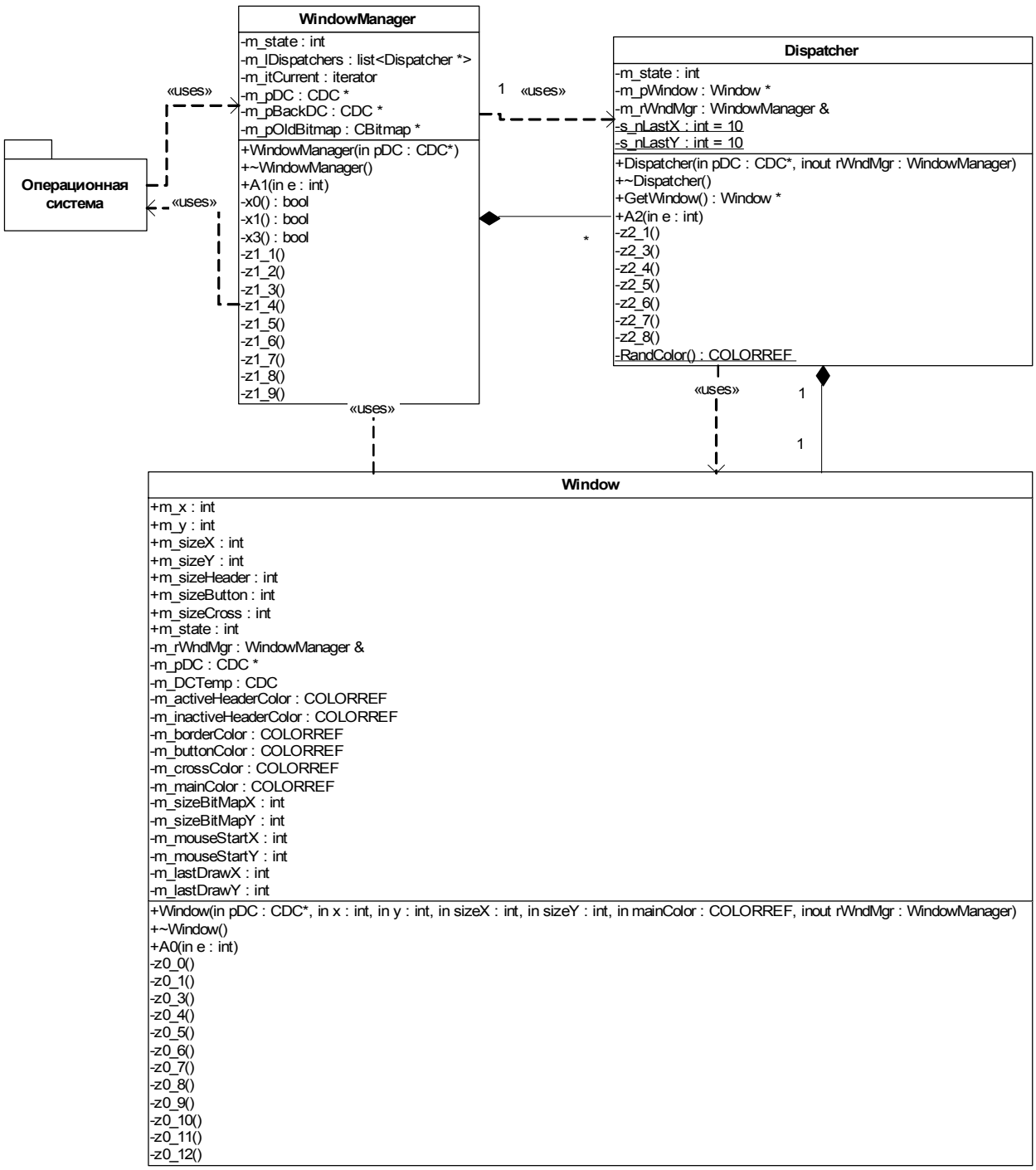

Рис. 3. Диаграмма классов

Пунктирные стрелки обозначают использование («uses») одного класса другим. К используемому классу подходит стрелка.

Сплошные линии с черным ромбом – агрегирование. Класс у ромба включает в себя класс на другом конце линии.

Обратим внимание на то, что объект класса “WindowManager” содержит множество (список) объектов класса “Dispatcher”.

3. Схема взаимодействий автоматов

Схема взаимодействий автоматов A0, A1, A2 и обработчиков событий приведена на рис. 4.

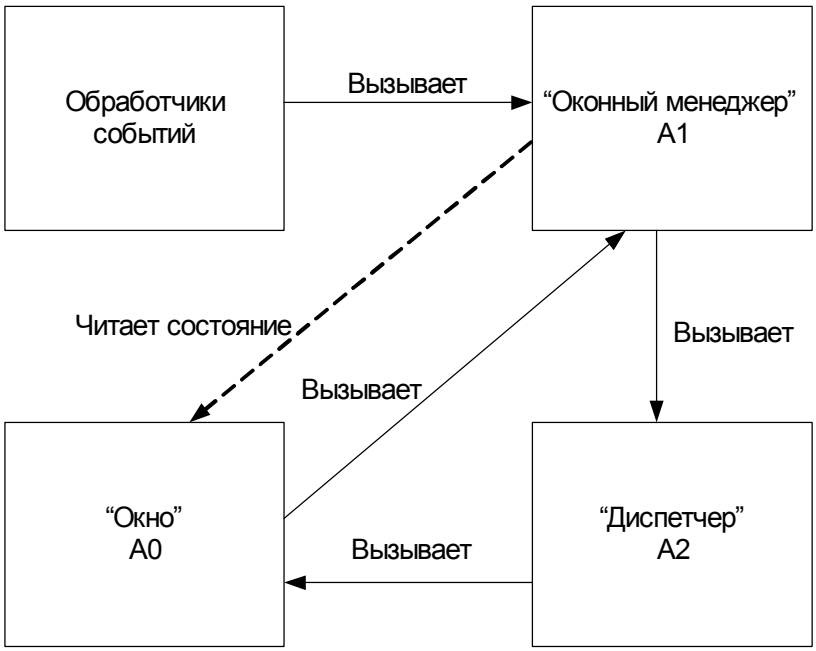

Рис. 4. Схема взаимодействий автоматов

Кажущейся, на первый взгляд, зацикленности не возникает. Цепочка вызовов обрывается в автомате “Окно”. Второй “виток” цикла не замыкается.

4. Класс “WindowManager”

4.1. Словесное описание

Класс “WindowManager” является оболочкой для автомата «Оконный менеджер». Он содержит список объектов класса "Dispatcher", каждому из которых соответствует одно окно.

4.2 Автомат «Оконный менеджер» (A1)

4.2.1. Словесное описание

Автомат управляет окнами. Он отвечает за получение и расшифровку событий от операционной системы. Также он обеспечивает выбор объекта из списка и вызов автомата "Диспетчер", соответствующего окну, которому предназначается событие. Например, автомат “Оконный менеджер” передает запросы о перерисовке всех окон или передвижении мыши.

4.2.2. Схема связей

Схема связей автомата «Оконный менеджер» (A1) представлена на рис. 5.

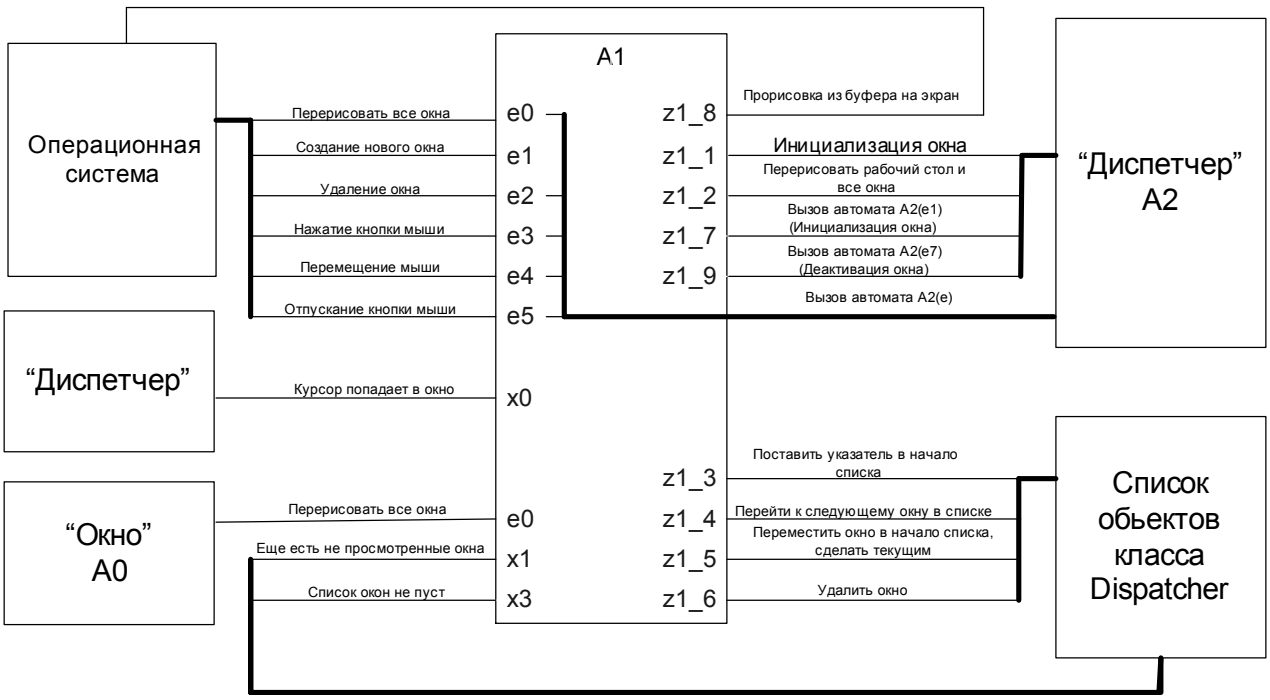

Рис. 5. Схема связей автомата "Оконный менеджер (A1)

4.2.3. Граф переходов

Граф переходов автомата «Оконный менеджер» (A1) представлен на рис. 6. Обратим внимание на не характерное для вызываемых автоматов пропускание через себя событий в состоянии 2. При этом автомат A2 вызывается не с внутренними событиями, формируемыми автоматом A1, а с теми событиями, с которыми вызван автомат A1. Автомат A1 является телом цикла *do – while*.

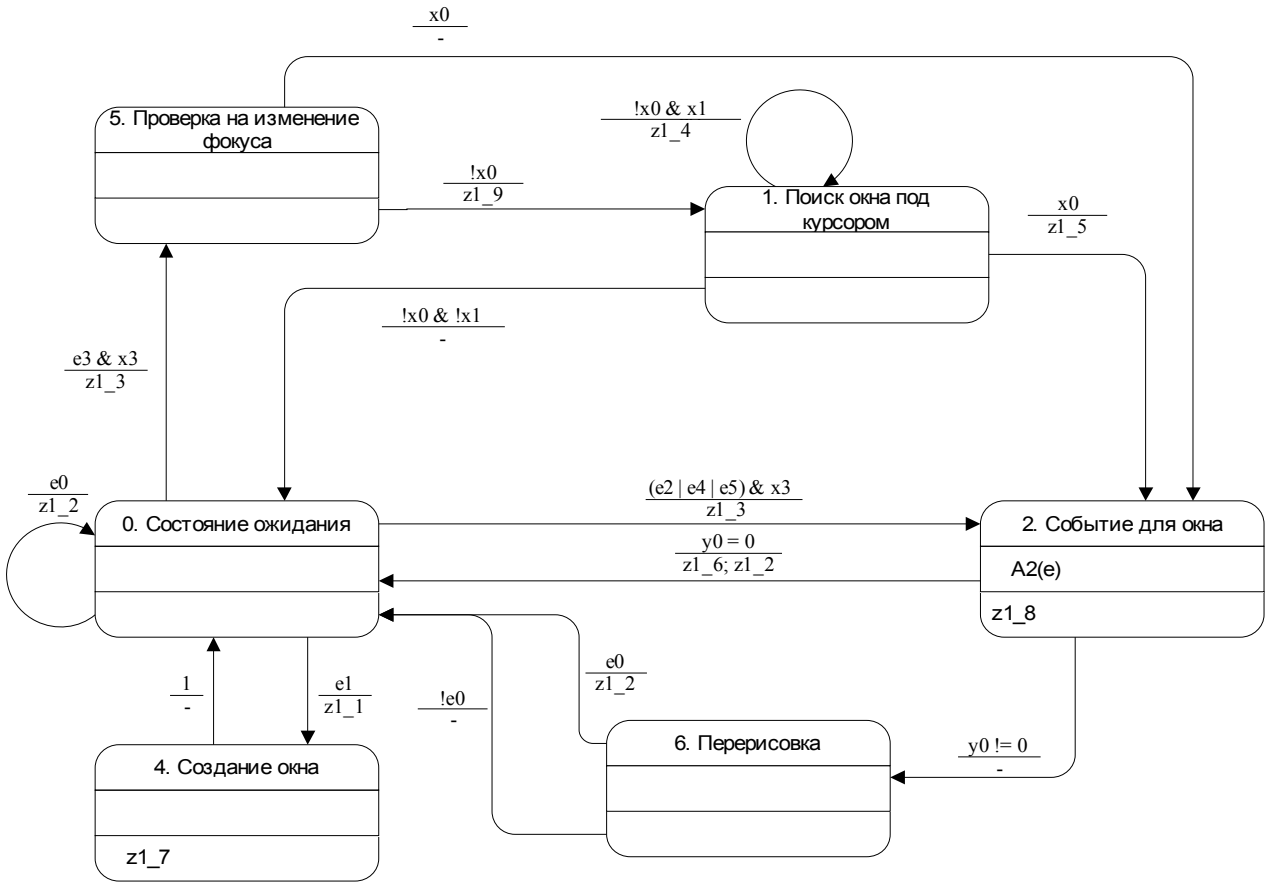

Рис. 6. Граф переходов автомата "Оконный менеджер (A1)"

5. Класс “Dispatcher”

5.1. Словесное описание

Класс “Dispatcher” является оболочкой для автомата «Диспетчер». Предоставляет автомату «Диспетчер» доступ ко всем необходимым данным об окне.

5.2 Автомат «Диспетчер» (A2)

5.2.1. Словесное описание

Каждому автомату «Диспетчер» соответствует один автомат «Окно», с которым автомат «Диспетчер» и работает. Он производит преобразование низкоуровневых событий (например, нажатие кнопки мыши в данной точке) в высокоуровневые события, передаваемые автомату «Окно». Таким образом, для реализации N окон необходимо иметь один «Оконный менеджер», N автоматов «Диспетчер» и N автоматов «Окно».

5.2.2. Схема связей

Схема связей автомата «Диспетчер» (A2) представлена на рис. 7.

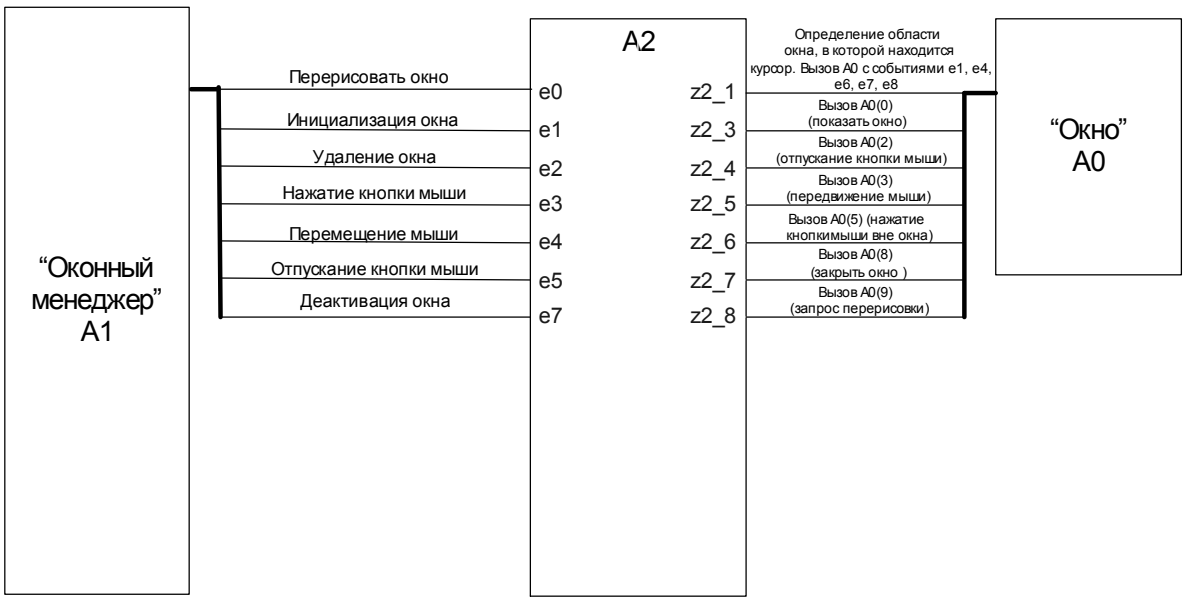

Рис. 7. Схема связей автомата "Диспетчер" (A2)

5.2.3. Граф переходов

Граф переходов автомата «Диспетчер» (A2) представлен на рис. 8.

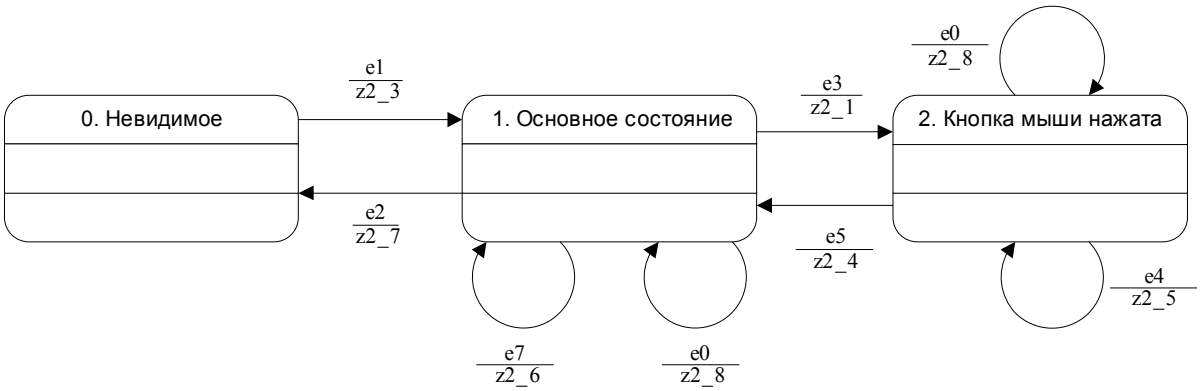

Рис. 8. Граф переходов автомата "Диспетчер" (A2)

6. Класс “Window”

6.1. Словесное описание

Класс “Window” является «оболочкой» для автомата «Окно». Он содержит все необходимое для непосредственного отображения окна на экран.

6.2 Автомат «Окно» (A0)

6.2.1. Словесное описание

Автомат «Окно» обеспечивает отрисовку, изменение размера, перемещение и минимизацию окна. Этот автомат обрабатывает высокоуровневые события, например, нажатие кнопки минимизации окна.

Выходные воздействия данного автомата управляют выводом на графическую подсистему операционной системы (Graphics Device Interface).

6.2.2. Схема связей

Схема связей автомата «Окно» представлена на рис. 9.

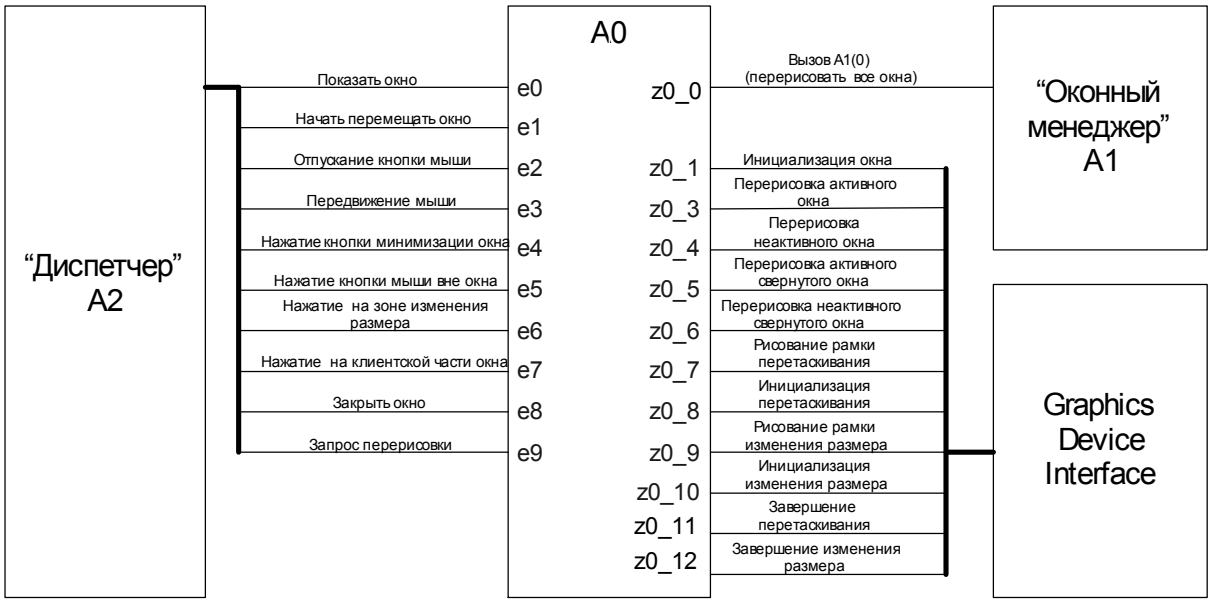

Рис. 9. Схема связей автомата "Окно" (A0)

6.2.3. Граф переходов

Граф переходов автомата «Окно» приведен на рис. 10.

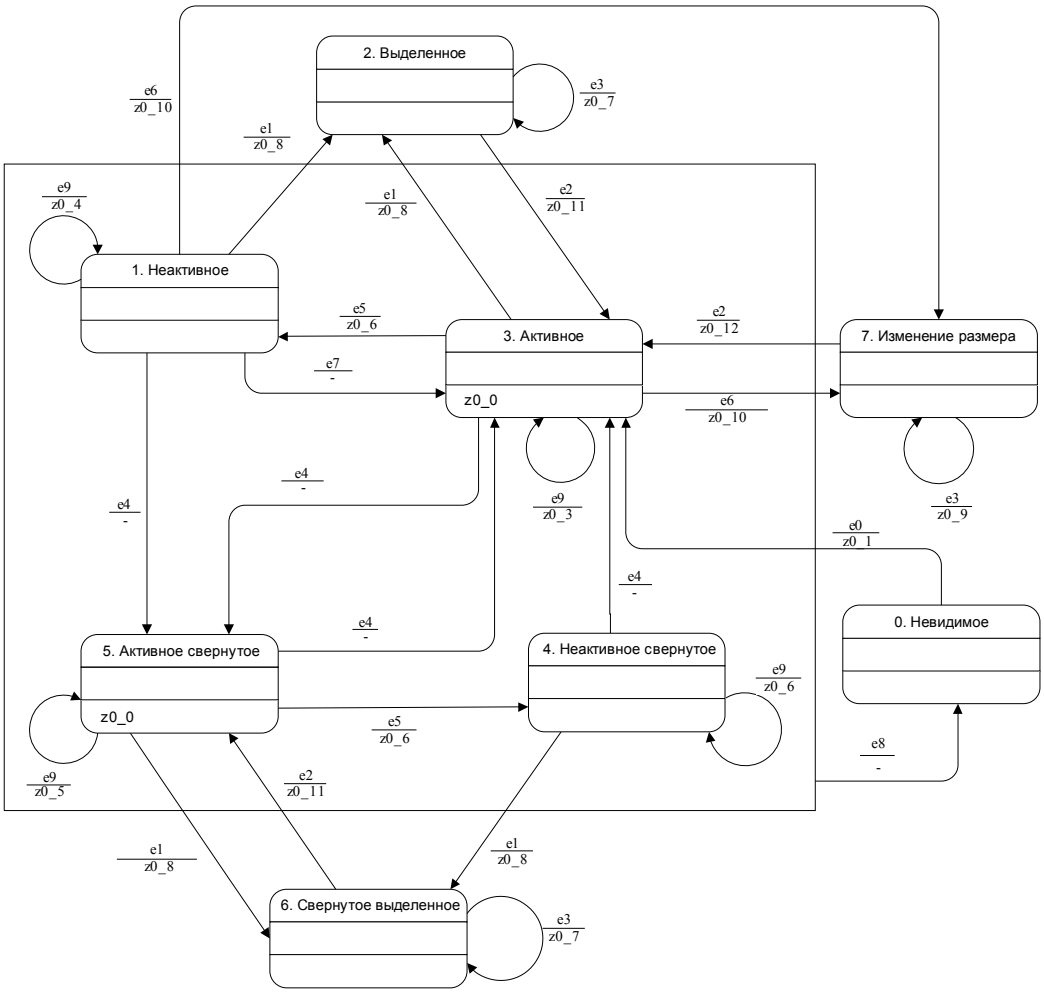

Рис. 10. Граф переходов автомата "Окно" (A0)

7. Листинги программы

Листинг 1. Файл "WindowManager.h"

```
#pragma once
#include "dispatcher.h"
#include <list>

class WindowManager
{

public:
    WindowManager(CDC *pDC);
    virtual ~WindowManager();

    // Функция автомата A1
    void A1(int e);

private:
    ////////////////////////////////////////////////////////////////////////
    // Входные переменные автомата
    bool x0()   const;
    bool x1()   const;
    bool x3()   const;

    ////////////////////////////////////////////////////////////////////////
    // Выходные воздействия автомата
    void z1_1();
    void z1_2();
    void z1_3();
    void z1_4();
    void z1_5();
    void z1_6();
    void z1_7();
    void z1_8();
    void z1_9();

    ////////////////////////////////////////////////////////////////////////
    // Состояние автомата

    int m_state;

    std::list<Dispatcher*> m_lDispatchers;
    std::list<Dispatcher*>::iterator m_itCurrent;
    CDC *m_pDC;
    CDC *m_pBackDC;
    CBitmap *m_pOldBitmap;
};
```

Листинг 2. Файл "WindowManager.cpp"

```
#include "stdafx.h"
#include "Win02.h"
#include "WindowManager.h"
#include "mainfrm.h"

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

//////////////////////////////////////////////////////////////////////
// Конструктор/Деструктор
//
WindowManager::WindowManager(CDC *pDC)
    :   m_pDC(pDC),
        m_pBackDC(new CDC())
        m_state(0)

{
    m_pBackDC->CreateCompatibleDC(m_pDC);
    CBitmap *bmp = new CBitmap();
    bmp->CreateCompatibleBitmap(m_pDC, m_pDC->GetDeviceCaps(HORZRES),
                                m_pDC->GetDeviceCaps(VERTRES));
    m_pOldBitmap = m_pBackDC->SelectObject(bmp);
}

WindowManager::~WindowManager()
{
    for(std::list<Dispatcher*>::iterator it = m_lDispatchers.begin();
                                        it  != m_lDispatchers.end(); ++it)
```

```
        delete (*it);

    delete m_pDC;
    delete m_pDC;
}

////////////////////////////////////////////////////////////////////
// Функция автомата A1
//
void WindowManager::A1(int e)
{
    do
    {
        int stateOld = m_state;

        switch(m_state)
        {
        case 0:                         // Состояние ожидания
            {
                  if(e == 0)            {   z1_2();                                   }
              else if(e == 1)           {   z1_1();                    m_state = 4;   }
              else if(e == 3 && x3())   {   z1_3();                    m_state = 5;   }
              else if((e == 2 || e == 4 || e == 5) && x3())
                                        {   z1_3();                    m_state = 2;   }
              break;
            }
        case 1:                         // Поиск окна под курсором
            {
                  if(!x0() && x1())     {   z1_4();                                   }
              else if(x0())             {   z1_5();                m_state = 2;    }
              else if(!x0() && !x1())   {                          m_state = 0;    }
              break;
            }
        case 2:                         // Событие для окна
            {
              int y0 = (*m_itCurrent)->GetWindow()->m_state;

                  if(y0 == 0)           {   z1_6(); z1_2();        m_state = 0;    }
              else if(y0 != 0)          {                          m_state = 6;    }
              break;
            }
        case 4:                         // Создание окна
            {
                                        {                          m_state = 0;    }
              break;
            }
        case 5:                         // Проверка на изменение фокуса
            {
                  if(!x0())             {   z1_9();                m_state = 1;    }
              else if(x0())             {                          m_state = 2;    }
              break;
            }
        case 6:                         // Проверка на необходимость перерисовки
            {


                   if(e == 0)           {   z1_2();                m_state = 0;    }
              else if(e != 0)           {                          m_state = 0;    }
              break;
            }
        }

        if(m_state == stateOld)
                    continue;

        switch(m_state)
        {
            case 0: {                                 break;  }     // Состояние ожидания
            case 1: {                                 break;  }     // Поиск окна под курсором
            case 2: {   (*m_itCurrent)->A2(e);
                        z1_8();                       break;  }     // Событие для окна
            case 4: {   z1_7();                       break;  }     // Создание окна
            case 5: {                                 break;  }     // Проверка на изменение
фокуса
            case 6: {                                 break;  }     // Перерисовка
        }
    }
    while(m_state != 0);
}


////////////////////////////////////////////////////////////////////
// Входные переменные автомата A1
//

bool WindowManager::x0()    const
{
    CPoint mouse = CMainFrame::getMousePos();
    bool result = false;
```

```
    if(     ((*m_itCurrent)->GetWindow()->m_x > mouse.x)
        ||  ((*m_itCurrent)->GetWindow()->m_y > mouse.y)
        ||  ((*m_itCurrent)->GetWindow()->m_x
                    + (*m_itCurrent)->GetWindow()->m_sizeX < mouse.x)
        ||  ((*m_itCurrent)->GetWindow()->m_y
                    + (*m_itCurrent)->GetWindow()->m_sizeY
                    + (*m_itCurrent)->GetWindow()->m_sizeHeader < mouse.y))
    {
        result = false;
    } else if(      ((*m_itCurrent)->GetWindow()->m_state == 1)
                ||  ((*m_itCurrent)->GetWindow()->m_state == 3)
                ||  (mouse.y < (*m_itCurrent)->GetWindow()->m_y
                                + (*m_itCurrent)->GetWindow()->m_sizeHeader))
    {
        result = true;
    }

    return result;
}


bool WindowManager::x1()    const
{
    std::list<Dispatcher*>::const_iterator it = m_itCurrent;
    return (++it != m_lDispatchers.end());
}

bool WindowManager::x3()    const
{
    return (m_lDispatchers.size() != 0);
}

////////////////////////////////////////////////////////////////////
// Выходные воздействия автомата A1
//

void WindowManager::z1_1()
{
    Dispatcher *pDispatcher = new Dispatcher(m_pBackDC, *this);
    if(m_lDispatchers.size() > 0)
        (*m_lDispatchers.begin())->A2(7);
    m_lDispatchers.push_front(pDispatcher);
    pDispatcher->A2(1);
    m_itCurrent = m_lDispatchers.begin();
}

void WindowManager::z1_2()
{
    m_pBackDC->FillRect(CRect(-10000, -10000, 10000, 10000), &CBrush());
    for(std::list<Dispatcher*>::reverse_iterator it = m_lDispatchers.rbegin();
                                                it  != m_lDispatchers.rend(); it++)
        (*it)->A2(0);

    m_pDC->BitBlt(0, 0, m_pDC->GetDeviceCaps(HORZRES),
                        m_pDC->GetDeviceCaps(VERTRES), m_pBackDC, 0, 0, SRCCOPY);
}

void WindowManager::z1_3()
{
    m_itCurrent = m_lDispatchers.begin();
}

void WindowManager::z1_4()
{
    m_itCurrent++;
}

void WindowManager::z1_5()
{
    Dispatcher *pDispatcher = *m_itCurrent;
    m_lDispatchers.erase(m_itCurrent);
    m_lDispatchers.push_front(pDispatcher);
    m_itCurrent = m_lDispatchers.begin();
}

void WindowManager::z1_6()
{
    delete *m_itCurrent;
    m_lDispatchers.erase(m_itCurrent);
}

void WindowManager::z1_7()
{
    (*m_itCurrent)->A2(1);
}
```

```
void WindowManager::z1_8()
{
    m_pDC->BitBlt(0, 0, m_pDC->GetDeviceCaps(HORZRES),
                        m_pDC->GetDeviceCaps(VERTRES), m_pBackDC, 0, 0, SRCCOPY);
}

void WindowManager::z1_9()
{
    (*m_itCurrent)->A2(7);
}
```

Листинг 3. Файл "Dispatcher.h"

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

class WindowManager;

class Dispatcher
{
public:
    Dispatcher(CDC *pDC, WindowManager &rWndMgr);
    virtual ~Dispatcher();

    Window *GetWindow()      { return m_pWindow; }

    // Функция автомата A2
    void A2(int e);

private:
    ////////////////////////////////////////////////////////////////////
    // Выходные воздействия автомата
    void z2_1();        // Определение области окна, в которую попадает курсор,
                        // вызов автомата A0 с событием, определяемым этой областью

    void z2_3();        // Вызов A0(e0)
    void z2_4();        // Вызов A0(e2)
    void z2_5();        // Вызов A0(e3)
    void z2_6();        // Вызов A0(e5)
    void z2_7();        // Вызов A0(e8)
    void z2_8();        // Вызов A0(e9)

    ////////////////////////////////////////////////////////////////////
    // Состояние автомата

    int m_state;

    ////////////////////////////////////////////////////////////////////
    Window *m_pWindow;
    WindowManager &m_rWndMgr;

    ////////////////////////////////////////////////////////////////////
    // Статические методы и свойства класса,
    // используемые при создании нового окна

    static COLORREF Dispatcher::RandColor();
    static int s_nLastX;
    static int s_nLastY;
};
```

Листинг 4. Файл "Dispatcher.cpp"

```
#include "stdafx.h"
#include "Win02.h"
#include "Dispatcher.h"
#include "WindowManager.h"
#include "mainfrm.h"

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

//////////////////////////////////////////////////////////////////////
// Статические методы и атрибуты
//
int Dispatcher::s_nLastX = 10;
int Dispatcher::s_nLastY = 10;
```

```
COLORREF Dispatcher::RandColor()
{
    return (COLORREF)((rand() & 0xff) | ((rand() & 0xff) << 8)
                                      | ((rand() & 0xff) << 16));
}


//////////////////////////////////////////////////////////////////////
// Конструктор/Деструктор
//
Dispatcher::Dispatcher(CDC *pDC, WindowManager &rWndMgr)
    :   m_pWindow(0),
        m_state(0),
        m_rWndMgr(rWndMgr)
{
    CSize size(pDC->GetDeviceCaps(HORZRES), pDC->GetDeviceCaps(VERTRES));

    m_pWindow = new Window(pDC, (s_nLastX += 50) % (size.cx - 300) + 50,
                           (s_nLastY += 50) % (size.cy - 250) + 50, 200, 150,
                          RandColor(), m_rWndMgr);
}

Dispatcher::~Dispatcher()
{
    delete m_pWindow;
}

//Автоматная функция
void Dispatcher::A2(int e)
{
    int stateOld = m_state;

    switch(m_state)
    {
    case 0:                         // Невидимое
        {
                if(e == 1)      {       z2_3();             m_state = 1;    }
            break;
        }
    case 1:                         // Основное состояние
        {
                if(e == 0)      {       z2_8();                             }
            else if(e == 2)     {       z2_7();             m_state = 0;    }
            else if(e == 3)     {       z2_1();             m_state = 2;    }
            else if(e == 7)     {       z2_6();                             }
            break;
        }
    case 2:                         // Кнопка мыши нажата
        {
                if(e == 0)      {       z2_8();                             }
            else if(e == 4)     {       z2_5();                             }
            else if(e == 5)     {       z2_4();             m_state = 1;    }
            break;
        }
    }
}

void Dispatcher::z2_1()
{
    CPoint mouse = CMainFrame::getMousePos();

    if( (m_pWindow->m_x > mouse.x) ||
        (m_pWindow->m_y > mouse.y) ||
        (m_pWindow->m_x + m_pWindow->m_sizeX < mouse.x) ||
        (m_pWindow->m_y + m_pWindow->m_sizeY + m_pWindow->m_sizeHeader < mouse.y))
    {
        return;
    }

    int diff1 = (m_pWindow->m_sizeHeader - m_pWindow->m_sizeButton) / 2;
    int diff2 = (m_pWindow->m_sizeButton - m_pWindow->m_sizeCross) / 2;

    if( (mouse.x >= m_pWindow->m_x + m_pWindow->m_sizeX - diff1
                                   - m_pWindow->m_sizeButton) &&
        (mouse.x <= m_pWindow->m_x + m_pWindow->m_sizeX - diff1) &&
        (mouse.y <= m_pWindow->m_y + diff1 + m_pWindow->m_sizeButton) &&
        (mouse.y >= m_pWindow->m_y + diff1))
    {
        m_pWindow->A0(8);
        return;
    }

    if( (mouse.x <= m_pWindow->m_x + m_pWindow->m_sizeX
                                   - diff1 * 2 - m_pWindow->m_sizeButton) &&
        (mouse.x >= m_pWindow->m_x + m_pWindow->m_sizeX
                                   - diff1 * 2 - m_pWindow->m_sizeButton * 2) &&
        (mouse.y <= m_pWindow->m_y + diff1 + m_pWindow->m_sizeButton) &&
        (mouse.y >= m_pWindow->m_y + diff1))
    {
```

```
        m_pWindow->A0(4);
        return;
    }

    if(mouse.y < m_pWindow->m_y + m_pWindow->m_sizeHeader)
    {
        m_pWindow->A0(1);
        return;
    }

    if( (mouse.x <= m_pWindow->m_x + m_pWindow->m_sizeX - diff1) &&
        (mouse.x >= m_pWindow->m_x + m_pWindow->m_sizeX - diff1
                                   - m_pWindow->m_sizeButton) &&
        (mouse.y <= m_pWindow->m_y + m_pWindow->m_sizeY - diff1
                                   + m_pWindow->m_sizeHeader) &&
        (mouse.y >= m_pWindow->m_y + m_pWindow->m_sizeY- diff1
                              - m_pWindow->m_sizeButton + m_pWindow->m_sizeHeader) &&
        ((m_pWindow->m_state == 3) || (m_pWindow->m_state == 1)))
    {
        m_pWindow->A0(6);
        return;
    }

    if((m_pWindow->m_state == 3) || (m_pWindow->m_state == 1))
    {
        m_pWindow->A0(7);
        return;
    }
}


void Dispatcher::z2_3()
{
    m_pWindow->A0(0);
}
void Dispatcher::z2_4()
{
    m_pWindow->A0(2);
}
void Dispatcher::z2_5()
{
    m_pWindow->A0(3);
}
void Dispatcher::z2_6()
{
    m_pWindow->A0(5);
}
void Dispatcher::z2_7()
{
    m_pWindow->A0(8);
}
void Dispatcher::z2_8()
{
    m_pWindow->A0(9);
}
```

Листинг 5. Файл "Window.h"

```
#pragma once

class WindowManager;

class Window
{
public:
    Window(CDC *pDC, int x, int y, int sizeX, int sizeY,
           COLORREF mainColor, WindowManager &rWndMgr);
    virtual ~Window();

    // Функция автомата A0
    void A0(int e);

    ////////////////////////////////////////////////////////////////////
    // Размеры элементов окна
    int m_x;
    int m_y;
    int m_sizeX;
    int m_sizeY;
    int m_sizeHeader;
    int m_sizeButton;
    int m_sizeCross;

    ////////////////////////////////////////////////////////////////////
    // Состояние автомата
    int m_state;
```

```
private:
    ////////////////////////////////////////////////////////////////////
    // Выходные воздействия автомата
    void z0_0();
    void z0_1();
    void z0_3();
    void z0_4();
    void z0_5();
    void z0_6();
    void z0_7();
    void z0_8();
    void z0_9();
    void z0_10();
    void z0_11();
    void z0_12();

    ////////////////////////////////////////////////////////////////////
    WindowManager &m_rWndMgr;

    ////////////////////////////////////////////////////////////////////
    // Служебные данные для прорисовки окна
    CDC *m_pDC;
    CDC m_DCTemp;

    COLORREF m_activeHeaderColor;
    COLORREF m_inactiveHeaderColor;
    COLORREF m_borderColor;
    COLORREF m_buttonColor;
    COLORREF m_crossColor;
    COLORREF m_mainColor;

    int m_sizeBitMapX;
    int m_sizeBitMapY;
    int m_mouseStartX;
    int m_mouseStartY;
    int m_lastDrawX;
    int m_lastDrawY;
};
```

Листинг 6. Файл "Window.cpp"

```
#include "stdafx.h"
#include "Window.h"
#include "windowmanager.h"
#include "mainfrm.h"


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

////////////////////////////////////////////////////////////////////
// Конструктор/Деструктор
//

Window::Window(CDC *pDC, int x, int y, int sizeX, int sizeY,
               COLORREF mainColor, WindowManager &rWndMgr)
    :   m_pDC(pDC),
        m_x(x),
        m_y(y),
        m_sizeX(sizeX),
        m_sizeY(sizeY),
        m_sizeHeader(22),
        m_sizeButton(16),
        m_sizeCross(12),
        m_mainColor(mainColor),
        m_state(0),
        m_rWndMgr(rWndMgr)
{
    m_activeHeaderColor     = COLORREF(RGB(100, 100, 100));
    m_inactiveHeaderColor   = COLORREF(RGB(200, 200, 200));
    m_borderColor           = COLORREF(RGB(  0,   0,   0));
    m_buttonColor           = COLORREF(RGB(  0,   0,   0));
    m_crossColor            = COLORREF(RGB(255, 255, 255));
    m_DCTemp.CreateCompatibleDC(m_pDC);
}

Window::~Window()
{
}
```

```
//////////////////////////////////////////////////////////////////
// Автоматная функция класса Window
//
void Window::A0(int e)
{
    int stateOld = m_state;

    switch(m_state)
    {
    case 0:                         // Невидимое
        {
              if(e == 0)            {   z0_1();                     m_state = 3;    }
            break;
        }
    case 1:                         // Неактивное
        {
              if(e == 1)            {   z0_8();                     m_state = 2;    }
            else if(e == 4)         {                               m_state = 5;    }
            else if(e == 6)         {   z0_10();                    m_state = 7;    }
            else if(e == 7)         {                               m_state = 3;    }
            else if(e == 8)         {                               m_state = 0;    }
            else if(e == 9)         {   z0_4();                                     }
            break;
        }
    case 2:                         // Выделенное
        {
              if(e == 2)            {   z0_11();                    m_state = 3;    }
            else if(e == 3)         {   z0_7();
}
            break;
        }
    case 3:                         // Активное
        {
              if(e == 1)            {   z0_8();                     m_state = 2;    }
            else if(e == 4)         {                               m_state = 5;    }
            else if(e == 5)         {   z0_6();                     m_state = 1;    }
            else if(e == 6)         {   z0_10();                    m_state = 7;    }
            else if(e == 8)         {                               m_state = 0;    }
            else if(e == 9)         {   z0_3();                                     }
            break;
        }
    case 4:                         // Неактивное свернутое
        {
              if(e == 1)            {   z0_8();                     m_state = 6;    }
            else if(e == 4)         {                               m_state = 3;    }
            else if(e == 8)         {                               m_state = 0;    }
            else if(e == 9)         {   z0_6();                                     }
            break;
        }
    case 5:                         // Активное свернутое
        {
              if(e == 1)            {   z0_8();                     m_state = 6;    }
            else if(e == 4)         {                               m_state = 3;    }
            else if(e == 5)         {   z0_6();                     m_state = 4;    }
            else if(e == 8)         {                               m_state = 0;    }
            else if(e == 9)         {   z0_5();                                     }
            break;
        }
    case 6:                         // Свернутое выделенное
        {
              if(e == 2)            {   z0_11();                    m_state = 5;    }
            else if(e == 3)         {   z0_7();                                     }

            break;
        }
    case 7:                         // Изменение размера
        {
              if(e == 2)            {   z0_12();                    m_state = 3;    }
            else if(e == 3)         {   z0_9();                                     }
            break;
        }

    }

    if(m_state == stateOld)
        return;

    switch(m_state)
    {
        case 0:     {                   break;  }       // Невидимое
        case 1:     {                   break;  }       // Неактивное
        case 2:     {                   break;  }       // Выделенное
        case 3:     {   z0_0();         break;  }       // Активное
        case 4:     {                   break;  }       // Неактивное свернутое
        case 5:     {   z0_0();         break;  }       // Активное свернутое
        case 6:     {                   break;  }       // Свернутое выделенное
        case 7:     {                   break;  }       // Изменение размера
    }
}
```

```
//////////////////////////////////////////////////////////////////////
// Выходные воздействия автомата A0
//

void Window::z0_0()
{
    m_rWndMgr.A1(0);
}

void Window::z0_1()
{
}


void Window::z0_3()
{
    CRect rect(m_x, m_y, m_x + m_sizeX, m_y + m_sizeHeader);
    m_pDC->FillRect(rect, &CBrush(m_borderColor));
    rect = CRect(m_x + 2 , m_y + 2, m_x + m_sizeX - 2, m_y + m_sizeHeader - 2);
    m_pDC->FillRect(rect, &CBrush(m_activeHeaderColor));


    int diff1 = (m_sizeHeader - m_sizeButton) / 2;
    rect = CRect(m_x + m_sizeX - diff1, m_y + diff1,
                 m_x + m_sizeX - diff1 - m_sizeButton, m_y + diff1 + m_sizeButton);
    m_pDC->FillRect(rect, &CBrush(m_buttonColor));

    int diff2 = (m_sizeButton - m_sizeCross) / 2;
    CPen pen(PS_SOLID, 1, m_crossColor);
    m_pDC->SelectObject(&pen);
    m_pDC->MoveTo(m_x + m_sizeX - diff1 - diff2 - m_sizeCross,
                  m_y + diff1 + diff2 + m_sizeCross);
    m_pDC->LineTo(m_x + m_sizeX - diff1 - diff2, m_y + diff1 + diff2);
    m_pDC->MoveTo(m_x + m_sizeX - diff1 - diff2 - m_sizeCross, m_y + diff1 + diff2);
    m_pDC->LineTo(m_x + m_sizeX - diff1 - diff2, m_y + diff1 + diff2 + m_sizeCross);

    rect = CRect(m_x + m_sizeX - diff1 * 2 - m_sizeButton, m_y + diff1,
                 m_x + m_sizeX - diff1 * 2 - m_sizeButton * 2,
                 m_y + diff1 + m_sizeButton);
    m_pDC->FillRect(rect, &CBrush(m_buttonColor));

    m_pDC->SelectObject(&pen);
    m_pDC->MoveTo(m_x + m_sizeX - diff1 * 2 - diff2 - m_sizeCross - m_sizeButton,
                  m_y + diff1 + diff2 + m_sizeCross);
    m_pDC->LineTo(m_x + m_sizeX - diff1 * 2 - diff2 - m_sizeButton,
                  m_y + diff1 + diff2 + m_sizeCross);


    rect = CRect(m_x, m_y + m_sizeHeader, m_x + m_sizeX, m_y + m_sizeHeader + m_sizeY);
    m_pDC->FillRect(rect, &CBrush(m_borderColor));
    rect = CRect(m_x + 2, m_y + m_sizeHeader + 2, m_x + m_sizeX - 2,
                 m_y + m_sizeHeader + m_sizeY - 2);
    m_pDC->FillRect(rect, &CBrush(m_mainColor));

    rect = CRect(m_x + m_sizeX - diff1, m_y + m_sizeY - diff1 + m_sizeHeader,
                 m_x + m_sizeX - diff1 - m_sizeButton,
                 m_y + m_sizeY- diff1 - m_sizeButton + m_sizeHeader);
    m_pDC->FillRect(rect, &CBrush(m_buttonColor));
    m_pDC->SelectObject(&pen);

    m_pDC->MoveTo(m_x + m_sizeX - diff1 - m_sizeButton + diff2,
                  m_y + m_sizeY - diff1 + m_sizeHeader - diff2);
    m_pDC->LineTo(m_x + m_sizeX - diff1 - diff2,
                  m_y + m_sizeY- diff1 - m_sizeButton + m_sizeHeader + diff2);
    m_pDC->MoveTo(m_x + m_sizeX - diff1 - m_sizeButton + diff2 + m_sizeCross / 2,
                  m_y + m_sizeY - diff1 + m_sizeHeader - diff2);
    m_pDC->LineTo(m_x + m_sizeX - diff1 - diff2,
                  m_y + m_sizeY- diff1 - m_sizeButton + m_sizeHeader
                      + diff2 + m_sizeCross / 2);
}

void Window::z0_4()
{
    CRect rect(m_x, m_y, m_x + m_sizeX, m_y + m_sizeHeader);
    m_pDC->FillRect(rect, &CBrush(m_borderColor));
    rect = CRect(m_x + 2 , m_y + 2, m_x + m_sizeX - 2, m_y + m_sizeHeader - 2);
    m_pDC->FillRect(rect, &CBrush(m_inactiveHeaderColor));


    int diff1 = (m_sizeHeader - m_sizeButton) / 2;
    rect = CRect(m_x + m_sizeX - diff1, m_y + diff1,
                 m_x + m_sizeX - diff1 - m_sizeButton, m_y + diff1 + m_sizeButton);
    m_pDC->FillRect(rect, &CBrush(m_buttonColor));

    int diff2 = (m_sizeButton - m_sizeCross) / 2;
    CPen pen(PS_SOLID, 1, m_crossColor);
    m_pDC->SelectObject(&pen);
    m_pDC->MoveTo(m_x + m_sizeX - diff1 - diff2 - m_sizeCross,
```

```
                m_y + diff1 + diff2 + m_sizeCross);
    m_pDC->LineTo(m_x + m_sizeX - diff1 - diff2, m_y + diff1 + diff2);
    m_pDC->MoveTo(m_x + m_sizeX - diff1 - diff2 - m_sizeCross, m_y + diff1 + diff2);
    m_pDC->LineTo(m_x + m_sizeX - diff1 - diff2, m_y + diff1 + diff2 + m_sizeCross);

    rect = CRect(m_x + m_sizeX - diff1 * 2 - m_sizeButton, m_y + diff1,
                 m_x + m_sizeX - diff1 * 2 - m_sizeButton * 2 ,
                 m_y + diff1 + m_sizeButton);
    m_pDC->FillRect(rect, &CBrush(m_buttonColor));

    m_pDC->SelectObject(&pen);
    m_pDC->MoveTo(m_x + m_sizeX - diff1 * 2 - diff2 - m_sizeCross - m_sizeButton,
                  m_y + diff1 + diff2 + m_sizeCross);
    m_pDC->LineTo(m_x + m_sizeX - diff1 * 2 - diff2 - m_sizeButton,
                  m_y + diff1 + diff2 + m_sizeCross);


    rect = CRect(m_x, m_y + m_sizeHeader, m_x + m_sizeX, m_y + m_sizeHeader + m_sizeY);
    m_pDC->FillRect(rect, &CBrush(m_borderColor));
    rect = CRect(m_x + 2, m_y + m_sizeHeader + 2, m_x + m_sizeX - 2,
                 m_y + m_sizeHeader + m_sizeY - 2);
    m_pDC->FillRect(rect, &CBrush(m_mainColor));

    rect = CRect(m_x + m_sizeX - diff1, m_y + m_sizeY - diff1 + m_sizeHeader,
                 m_x + m_sizeX - diff1 - m_sizeButton,
                 m_y + m_sizeY- diff1 - m_sizeButton + m_sizeHeader);
    m_pDC->FillRect(rect, &CBrush(m_buttonColor));
    m_pDC->SelectObject(&pen);
    m_pDC->MoveTo(m_x + m_sizeX - diff1 - m_sizeButton + diff2,
                  m_y + m_sizeY - diff1 + m_sizeHeader - diff2);
    m_pDC->LineTo(m_x + m_sizeX - diff1 - diff2,
                  m_y + m_sizeY- diff1 - m_sizeButton + m_sizeHeader + diff2);
    m_pDC->MoveTo(m_x + m_sizeX - diff1 - m_sizeButton + diff2 + m_sizeCross / 2,
                  m_y + m_sizeY - diff1 + m_sizeHeader - diff2);
    m_pDC->LineTo(m_x + m_sizeX - diff1 - diff2,
                  m_y + m_sizeY- diff1 - m_sizeButton + m_sizeHeader
                      + diff2 + m_sizeCross / 2);
}

void Window::z0_5()
{
    CRect rect(m_x, m_y, m_x + m_sizeX, m_y + m_sizeHeader);
    m_pDC->FillRect(rect, &CBrush(m_borderColor));
    rect = CRect(m_x + 2 , m_y + 2, m_x + m_sizeX - 2, m_y + m_sizeHeader - 2);
    m_pDC->FillRect(rect, &CBrush(m_activeHeaderColor));


    int diff1 = (m_sizeHeader - m_sizeButton) / 2;
    rect = CRect(m_x + m_sizeX - diff1, m_y + diff1,
                 m_x + m_sizeX - diff1 - m_sizeButton, m_y + diff1 + m_sizeButton);
    m_pDC->FillRect(rect, &CBrush(m_buttonColor));

    int diff2 = (m_sizeButton - m_sizeCross) / 2;
    CPen pen(PS_SOLID, 1, m_crossColor);
    m_pDC->SelectObject(&pen);
    m_pDC->MoveTo(m_x + m_sizeX - diff1 - diff2 - m_sizeCross,
                  m_y + diff1 + diff2 + m_sizeCross);
    m_pDC->LineTo(m_x + m_sizeX - diff1 - diff2, m_y + diff1 + diff2);
    m_pDC->MoveTo(m_x + m_sizeX - diff1 - diff2 - m_sizeCross, m_y + diff1 + diff2);
    m_pDC->LineTo(m_x + m_sizeX - diff1 - diff2, m_y + diff1 + diff2 + m_sizeCross);

    rect = CRect(m_x + m_sizeX - diff1 * 2 - m_sizeButton, m_y + diff1,
                 m_x + m_sizeX - diff1 * 2 - m_sizeButton * 2 ,
                 m_y + diff1 + m_sizeButton);
    m_pDC->FillRect(rect, &CBrush(m_buttonColor));

    m_pDC->SelectObject(&pen);
    m_pDC->MoveTo(m_x + m_sizeX - diff1 * 2 - diff2 - m_sizeCross - m_sizeButton,
                  m_y + diff1 + diff2 + m_sizeCross);
    m_pDC->LineTo(m_x + m_sizeX - diff1 * 2 - diff2 - m_sizeButton,
                  m_y + diff1 + diff2 + m_sizeCross);
}

void Window::z0_6()
{
    CRect rect(m_x, m_y, m_x + m_sizeX, m_y + m_sizeHeader);
    m_pDC->FillRect(rect, &CBrush(m_borderColor));
    rect = CRect(m_x + 2 , m_y + 2, m_x + m_sizeX - 2, m_y + m_sizeHeader - 2);
    m_pDC->FillRect(rect, &CBrush(m_inactiveHeaderColor));


    int diff1 = (m_sizeHeader - m_sizeButton) / 2;
    rect = CRect(m_x + m_sizeX - diff1, m_y + diff1,
                 m_x + m_sizeX - diff1 - m_sizeButton, m_y + diff1 + m_sizeButton);
    m_pDC->FillRect(rect, &CBrush(m_buttonColor));

    int diff2 = (m_sizeButton - m_sizeCross) / 2;
    CPen pen(PS_SOLID, 1, m_crossColor);
    m_pDC->SelectObject(&pen);
```

```
    m_pDC->MoveTo(m_x + m_sizeX - diff1 - diff2 - m_sizeCross,
                  m_y + diff1 + diff2 + m_sizeCross);
    m_pDC->LineTo(m_x + m_sizeX - diff1 - diff2, m_y + diff1 + diff2);
    m_pDC->MoveTo(m_x + m_sizeX - diff1 - diff2 - m_sizeCross, m_y + diff1 + diff2);
    m_pDC->LineTo(m_x + m_sizeX - diff1 - diff2, m_y + diff1 + diff2 + m_sizeCross);

    rect = CRect(m_x + m_sizeX - diff1 * 2 - m_sizeButton, m_y + diff1,
                 m_x + m_sizeX - diff1 * 2 - m_sizeButton * 2 ,
                 m_y + diff1 + m_sizeButton);
    m_pDC->FillRect(rect, &CBrush(m_buttonColor));

    m_pDC->SelectObject(&pen);
    m_pDC->MoveTo(m_x + m_sizeX - diff1 * 2 - diff2 - m_sizeCross - m_sizeButton,
                  m_y + diff1 + diff2 + m_sizeCross);
    m_pDC->LineTo(m_x + m_sizeX - diff1 * 2 - diff2 - m_sizeButton,
                  m_y + diff1 + diff2 + m_sizeCross);
}

void Window::z0_7()
{
    m_pDC->BitBlt(m_lastDrawX, m_lastDrawY, m_sizeBitMapX, m_sizeBitMapY,
                  &m_DCTemp, 0, 0, SRCCOPY);

    CBitmap bmp;
    bmp.CreateCompatibleBitmap(m_pDC, m_sizeBitMapX, m_sizeBitMapY);
    m_DCTemp.SelectObject(&bmp);

    CPoint mouse = CMainFrame::getMousePos();

    m_DCTemp.BitBlt(0, 0, m_sizeBitMapX, m_sizeBitMapY, m_pDC,
                    m_x + mouse.x - m_mouseStartX - 2,
                    m_y + mouse.y - m_mouseStartY - 2, SRCCOPY);
    CPen pen(PS_DASH, 1, m_borderColor);

    m_pDC->SelectObject(&pen);

    m_pDC->MoveTo(m_x + mouse.x - m_mouseStartX, m_y + mouse.y - m_mouseStartY);
    m_pDC->LineTo(m_x + mouse.x - m_mouseStartX + m_sizeX,
                  m_y + mouse.y - m_mouseStartY);
    m_pDC->LineTo(m_x + mouse.x - m_mouseStartX + m_sizeX,
                  m_y + mouse.y - m_mouseStartY + m_sizeY + m_sizeHeader);
    m_pDC->LineTo(m_x + mouse.x - m_mouseStartX,
                  m_y + mouse.y - m_mouseStartY + m_sizeY + m_sizeHeader);
    m_pDC->LineTo(m_x + mouse.x - m_mouseStartX, m_y + mouse.y - m_mouseStartY);

    m_lastDrawX = m_x + mouse.x - m_mouseStartX - 2;
    m_lastDrawY = m_y + mouse.y - m_mouseStartY - 2 ;
}

void Window::z0_8()
{
    m_sizeBitMapX = m_sizeX + 15;
    m_sizeBitMapY = m_sizeY + m_sizeHeader + 15;
    CBitmap bmp;
    bmp.CreateCompatibleBitmap(m_pDC, m_sizeBitMapX, m_sizeBitMapY);
    m_DCTemp.SelectObject(&bmp);
    m_DCTemp.BitBlt(0, 0, m_sizeBitMapX, m_sizeBitMapY, m_pDC,
                    m_x - 2, m_y - 2, SRCCOPY );
    m_mouseStartX = CMainFrame::getMousePos().x;
    m_mouseStartY = CMainFrame::getMousePos().y;
    m_lastDrawX = m_x - 2;
    m_lastDrawY = m_y - 2;
}

void Window::z0_9()
{
    m_pDC->BitBlt(m_x - 2, m_y - 2, m_sizeBitMapX,
                  m_sizeBitMapY, &m_DCTemp, 0, 0, SRCCOPY);
    if(((CMainFrame::getMousePos().x - m_x) >= m_sizeButton)
        && ((CMainFrame::getMousePos().y - m_y) >= m_sizeHeader))
    {
        m_sizeBitMapX = (CMainFrame::getMousePos().x - m_x) + 15;
        m_sizeBitMapY = (CMainFrame::getMousePos().y - m_y) + 15;

        CBitmap bmp;
        bmp.CreateCompatibleBitmap(m_pDC, m_sizeBitMapX, m_sizeBitMapY);
        m_DCTemp.SelectObject(&bmp);

        m_DCTemp.BitBlt(0, 0, m_sizeBitMapX, m_sizeBitMapY, m_pDC,
                        m_x - 2, m_y - 2, SRCCOPY);
        CPen pen(PS_DASH, 1, m_borderColor);

        m_pDC->SelectObject(&pen);
        m_pDC->MoveTo(m_x, CMainFrame::getMousePos().y);
        m_pDC->LineTo(CMainFrame::getMousePos().x, CMainFrame::getMousePos().y);
        m_pDC->MoveTo(CMainFrame::getMousePos().x, m_y);
        m_pDC->LineTo(CMainFrame::getMousePos().x, CMainFrame::getMousePos().y);
        m_pDC->MoveTo(m_x, m_y);
        m_pDC->LineTo(CMainFrame::getMousePos().x, m_y);
```

```
        m_pDC->MoveTo(m_x, m_y);
        m_pDC->LineTo(m_x, CMainFrame::getMousePos().y);
    }
}

void Window::z0_10()
{
    m_sizeBitMapX = m_sizeX + 15;
    m_sizeBitMapY = m_sizeY + m_sizeHeader + 15;

    CBitmap bmp;
    bmp.CreateCompatibleBitmap(m_pDC, m_sizeBitMapX, m_sizeBitMapY);
    m_DCTemp.SelectObject(&bmp);


    m_DCTemp.BitBlt(0, 0, m_sizeBitMapX, m_sizeBitMapY, m_pDC,
                    m_x - 2, m_y - 2, SRCCOPY );
}

void Window::z0_11()
{
    m_x += CMainFrame::getMousePos().x - m_mouseStartX;
    m_y += CMainFrame::getMousePos().y - m_mouseStartY;
}

void Window::z0_12()
{
    m_sizeX = CMainFrame::getMousePos().x - m_x;
    m_sizeY = CMainFrame::getMousePos().y - m_y - m_sizeHeader;
    if(m_sizeX < m_sizeButton * 3)
    {
        m_sizeX = m_sizeButton * 3;
    }
    if(m_sizeY < m_sizeButton * 1.5)
    {
        m_sizeY = m_sizeButton * 1.5;
    }
}
```