Санкт-Петербургский государственный университет
информационных технологий, механики и оптики

Кафедра «Компьютерные технологии»

В.А. Добровольский, А.В. Степук

# ПРОСТОЙ АУДИОПРОИГРЫВАТЕЛЬ

Программирование с явным выделением состояний

Проектная документация

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

Санкт-Петербург
2004

## Оглавление

# Введение

Известны аудиопроигрыватели, например, *Windows Media Player* описанный в работе [1]. Однако, в характерном для компании *Microsoft* стиле исходный код и документация на проект не приводятся. Для других проектов, например [2], характерно наличие открытого исходного кода при отсутствии проектной документации.

Цель настоящей работы состоит в разработке простейшего аудиопроигрывателя, для которого характерно:

- построение его системы управления на основе автоматного подхода [3];
- разработка открытой проектной документации [4].

Достоинства автоматного подхода состоят в следующем. Во-первых, его использование позволяет повысить централизацию логики управления в программном коде, а во-вторых, код является изоморфным графу переходов, по которому он строился. Кроме того, проектирование вынесено в отдельный этап, который и является основной частью создания программы.

Проект создан в рамках «Движения за открытую проектную документацию». В проекте основное внимание уделяется формализации построения управляющей части программы. Проект реализован на языке *C++* с использованием библиотеки *MFC* в среде программирования *Microsoft Visual C++ 6.0.*

# 1. Постановка задачи

Разработать программу, используя автоматный подход, которая реализует аудиопроигрыватель с минимальной необходимой функциональностью.

При этом работа состоит из следующих этапов:

- выбор интерфейса аудиопроигрывателя;
- разработка управляющего автомата;
- программная реализация автомата;
- построение программы аудиопроигрывателя.

# 2. Интерфейс аудиопроигрывателя

На рис. 1 приведен интерфейс разрабатываемого устройства.


**Рис. 1.** Внешний вид устройства

Окно состоит из двух областей:

- кнопки;
- индикатор.

Кнопки имеют следующие назначения:

- (*play*) – начать проигрывание;
- (*stop)* – остановить проигрывание;
- (*pause)* – сделать паузу, а при повторном нажатии – продолжить проигрывание;
- (*prev)* – перейти на предыдущий трек;
- (*next)* – перейти на следующий трек.

Индикатор отображает название музыкального файла, номер трека и текущее состояние. Данное устройство может находиться в шести состояниях:

1. Готов к работе (Ready).
2. Нет треков (NO TRACK).
3. Играет трек (Playing).
4. Прокрутка трека вперед
5. Прокрутка трека назад
6. Пауза (Pause).

Состояние 4 и 5 не визуализируются, потому что эти состояния являются неустойчивыми.

## 3. Диаграмма классов

Диаграмма классов приведена на рис. 2.

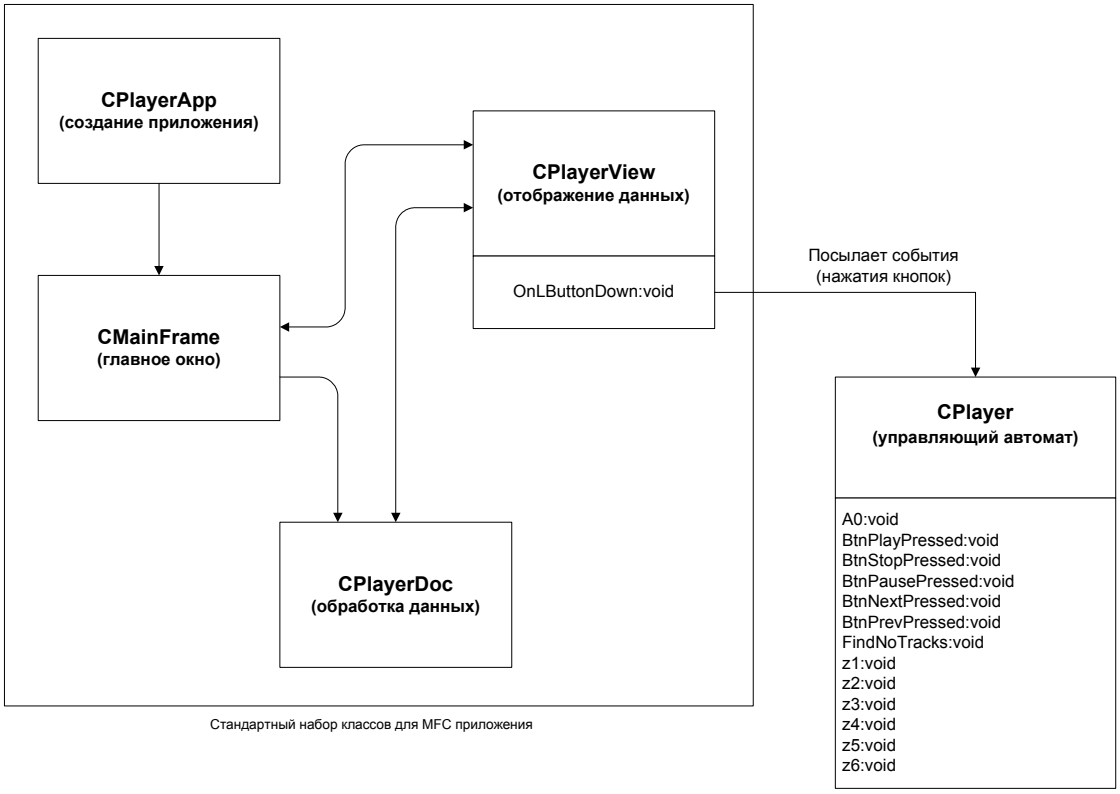


**Рис. 2.** Диаграмма классов

Эта диаграмма содержит пять классов:

1. *CPlayerApp* – создание приложения.
2. *CMainFram*e - главное окно.
3. *CPlayerView* – отображение данных.
4. *CPlayerDoc* – обработка данных.
5. *CPlayer* – управляющий автомат.

# 4. Проектирование автомата

Система управления аудиопроигрывателя представляет собой автомат *A0*. Нажатия кнопок рассматриваются как события, воздействующие на автомат. Выходными воздействиями являются операции перехода треков, проигрывания трека, управления дисплеем и т.д. Интерфейс автомата отображается его схемой связей (рис. 3).

Класс *CPlayer* реализует автомат без использования механизма наследования.

Функция *A0*(int *e*) реализует логику изменения состояний автомата. Она построена по его графу переходов (рис. 4). Эта функция реализована с помощью двух операторов *switch*, в первом из которых выполняются переходы, а во втором вызываются выходные воздействия при переходе в это состояние.

Обозначения *A0*(0) соответствует инициализация автомата после отработки функций выходных воздействий *z3*() и *z4*(). Это объясняется тем, что

после перемотки трека необходимо инициализировать автомат и запустить трек на проигрывание.

Функции *BtnPlayPressed*(), *BtnStopPressed*(), *BtnPausePressed*(), *BtnNextPressed*(), *BtnPrevPressed*(), *FindNoTracks*() обрабатывают события, приходящие от пользователя, и вызывают функцию *A0*(int *e*).

Функции *z1*(), … , *z6*() реализуют выходные воздействия автомата.

## 4.1. Схема связей автомата

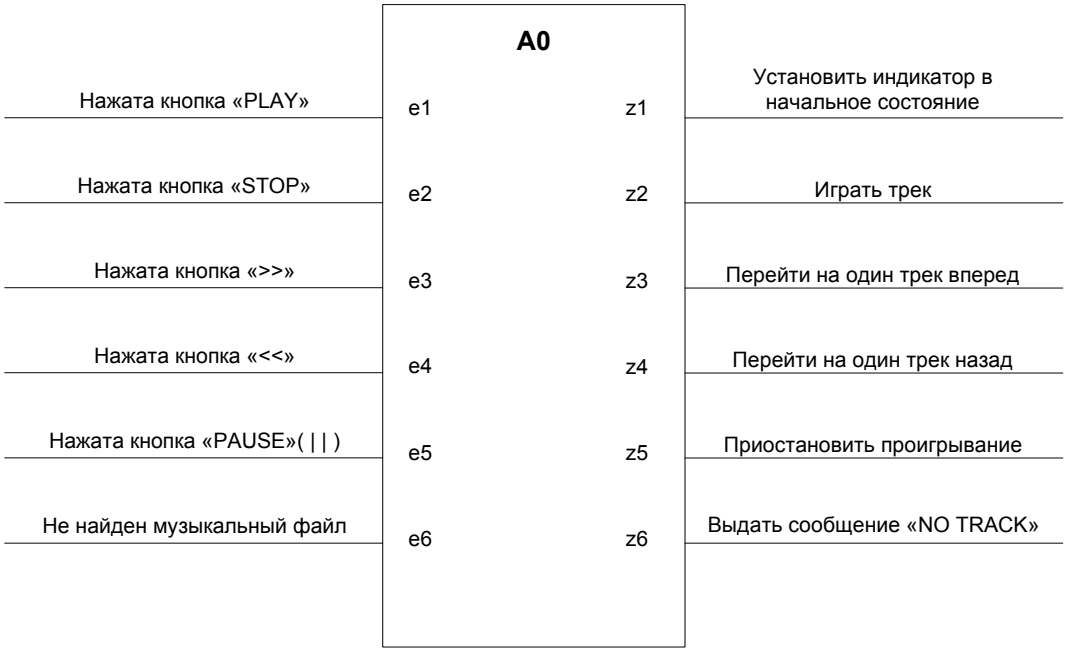


**Рис. 3.** Схема связей автомата

## 4.2. Граф переходов

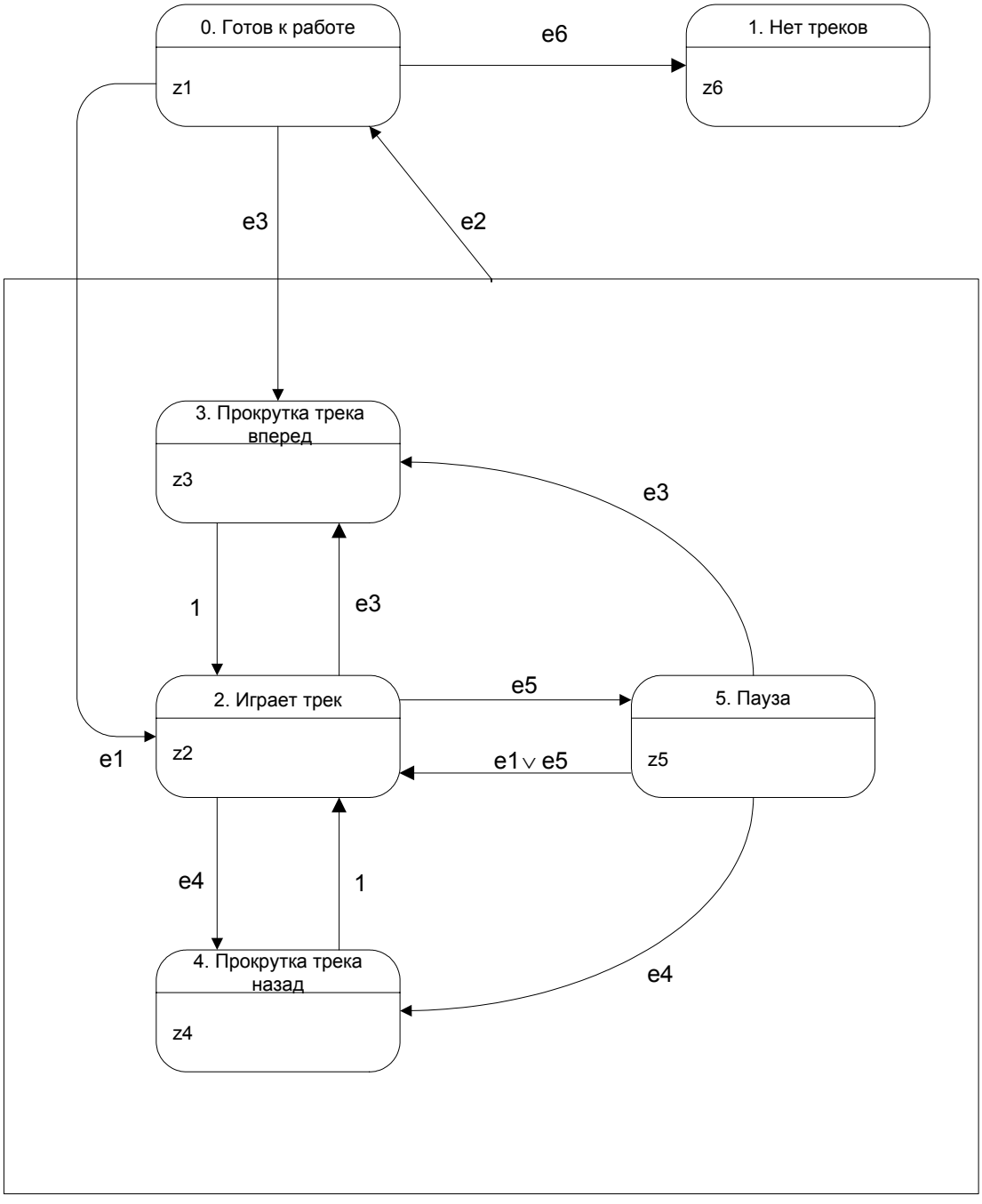


**Рис. 4.** Граф переходов автомата

# 5. Исходный код автомата

```
void CPlayer::A0(int e)
{
	int yold = y0;

	switch(y0)
	{
	case 0: // "Готов к работе"
		if(e == 3) { y0 = 3; }
		if(e == 6) { y0 = 1; }
		if(e == 1) { y0 = 2; }
		break;

	case 1: // "Нет треков"
		break;

	case 2: // "Играть трек"
		if(e == 5) { y0 = 5; }
		if(e == 4) { y0 = 4; }
		if(e == 3) { y0 = 3; }
		if(e == 2) { y0 = 0; }
		break;

	case 3: // "Прокрутка трека вперед"
		{ y0 = 2; }
		break;

	case 4: // "Прокрутка трека назад"
		{ y0 = 2; }
		break;

	case 5: // "Пауза"
		if (e == 5)	{ y0 = 2; }
		if (e == 1)	{ y0 = 2; }
		if (e == 2)	{ y0 = 0; }
		if (e == 3)	{ y0 = 3; }
		if (e == 4)	{ y0 = 4; }
		break;
	}

	if(y0 != yold)
	{
		switch(y0)
		{
		case 0:
			{ z1(); }
			break;

		case 1:
			{ z6(); }
			break;
```

```
                case 2:
                        { z2(); }
                        break;

                case 3:
                        { z3();}
                        break;

                case 4:
                        { z4(); }
                        break;

                case 5:
                        { z5(); }
                        break;
                }
        }
}
```

# 6. Основные фрагменты программы

Ниже приводятся фрагменты программы, созданные авторами. Остальная часть программы генерируется автоматически.

### 1. Фрагменты исходного кода класса *CPlayer*

**// Construction/Destruction**

```
CPlayer* CPlayer::_instance = 0;

int nTrackNumber = 1;

CPlayer::CPlayer():
m_timeTrack(0),
m_strTitle("NO DISK"),
y0(0),
m_strFile("mus/1.wav")
{
}

CPlayer::~CPlayer()
{
}
```

**// Отрисовка внешнего пользовательского интерфейса**

```
void CPlayer::OnDraw(CDC *pDC)
{
        CMainFrame *pFrame = (CMainFrame*)AfxGetApp()->m_pMainWnd;
        CPlayerView *pView = (CPlayerView*) pFrame->GetActiveView();
        CRect rect(CPoint(0,0), m_size);
        pDC->FillSolidRect(rect, RGB(200,200,250));
        CPen *pen_old = pDC->GetCurrentPen();
        CPen *pen_black = new CPen(PS_SOLID, 1, RGB(0,0,0));
        CPen *pen_red = new CPen(PS_SOLID, 2, RGB(255,0,0));
        pDC->SelectObject(pen_black);
        for (int i = 0; i < 5; i++)
```

```
{
        rect.right = i*(m_size.cx+0.5)/5;
        rect.left = rect.right + (m_size.cx+0.5)/5;
        rect.bottom = m_size.cy-3;
        rect.top = rect.bottom - m_size.cy/8;

        rect.InflateRect(4, 2, 4, 2);
        pDC->Rectangle(rect);
        rect.InflateRect(3, -3, 3, -3);
        pDC->FillSolidRect(rect, RGB(100,100,150));
        pDC->SelectObject(pen_red);
        rect.NormalizeRect();
        CPoint pt(rect.Width()/2, rect.Height()/2);
        pt.x = rect.left + rect.Width()/2;
        pt.y = rect.top + rect.Height()/2;
        pDC->MoveTo(pt);

        // Отрисовка кнопок
        switch(i)
        {
        case 0: //play
                pt.y -=4;
                pDC->LineTo(pt);
                pt.x +=8; pt.y += 4;
                pDC->LineTo(pt);
                pt.x -=8; pt.y += 4;
                pDC->LineTo(pt);
                pt.y -= 4;
                pDC->LineTo(pt);
                break;
        case 1: //stop
                pt.y -=4;
                pDC->LineTo(pt);
                pt.x +=8; pt.y += 0;
                pDC->LineTo(pt);
                pt.x -=0; pt.y += 8;
                pDC->LineTo(pt);
                pt.x -=8; pt.y += 0;
                pDC->LineTo(pt);
                pt.y -= 4;
                pDC->LineTo(pt);
                break;
        case 2: //pause
                pt.x -= 4; pt.y -=4;
                pDC->MoveTo(pt);
                pt.y += 8;
                pDC->LineTo(pt);
                pt.x += 8;
                pDC->MoveTo(pt);
                pt.y -= 8;
                pDC->LineTo(pt);
                break;
        case 3: //prev
                pt.x += 8;
                pDC->MoveTo(pt);
                pt.y -=4;
                pDC->LineTo(pt);
```

```
                        pt.x -=8; pt.y += 4;
                        pDC->LineTo(pt);
                        pt.x +=8; pt.y += 4;
                        pDC->LineTo(pt);
                        pt.y -= 4;
                        pDC->LineTo(pt);

                        pt.x -= 8;
                        pDC->MoveTo(pt);
                        pt.y -=4;
                        pDC->LineTo(pt);
                        pt.x -=8; pt.y += 4;
                        pDC->LineTo(pt);
                        pt.x +=8; pt.y += 4;
                        pDC->LineTo(pt);
                        pt.y -= 4;
                        pDC->LineTo(pt);
                        break;
                case 4: //next
                        pt.x -= 8;
                        pDC->MoveTo(pt);
                        pt.y -=4;
                        pDC->LineTo(pt);
                        pt.x +=8; pt.y += 4;
                        pDC->LineTo(pt);
                        pt.x -=8; pt.y += 4;
                        pDC->LineTo(pt);
                        pt.y -= 4;
                        pDC->LineTo(pt);

                        pt.x += 8;
                        pDC->MoveTo(pt);
                        pt.y -=4;
                        pDC->LineTo(pt);
                        pt.x +=8; pt.y += 4;
                        pDC->LineTo(pt);
                        pt.x -=8; pt.y += 4;
                        pDC->LineTo(pt);
                        pt.y -= 4;
                        pDC->LineTo(pt);
                        break;
                }
                pDC->SelectObject(pen_old);
        }
        CRect rect_view(m_size.cx / 16, m_size.cy / 16, m_size.cx - m_size.cx / 16 ,
m_size.cy - m_size.cy / 3);
        pDC->FillSolidRect(rect_view, RGB(10,10,10));

        delete pen_black;
        delete pen_red;

        DrawTitles(rect_view, pDC);
}

CPlayer *const CPlayer::Instance()
{
        if (_instance == 0) _instance = new CPlayer ;
```

```
	return _instance;
}

void CPlayer::DeleteClass()
{
	delete _instance;
	_instance = 0;
}

void CPlayer::SetNewSize(CSize size)
{
	m_size = size;
CPlayerView *pView = (CPlayerView*) pFrame->GetActiveView();
}

CSize CPlayer::GetSize() const
{
	return m_size;
}

void CPlayer::DrawTitles(CRect rect, CDC *pDC)
{
	CFont fntTime, fntTrack;
	fntTime.CreateFont(rect.Height()/3, 0, 0, 0, FW_NORMAL, 0, 0, 0, 0, 0, 0, 0, 0,
"Courier");
	fntTrack.CreateFont(rect.Height()/5, 0, 0, 0, FW_NORMAL, 0, 0, 0, 0, 0, 0, 0, 0,
"Arial");
	pDC->SetTextColor(RGB(50, 250, 50));
	pDC->SelectObject(&fntTime);
	pDC->DrawText(m_strFile.c_str(), rect, 0);
	pDC->SelectObject(&fntTrack);
	CRect rect_title = rect;
	rect_title.top += rect_title.Height()*2/3;
	pDC->DrawText(m_strTitle, rect_title, 0);
}
```

**// Выходные воздействия**

```
void CPlayer::z1()
{
	m_timeTrack = 0;

	nTrackNumber = 1;
	//i_Sound.StopSoundFile();
	m_strTitle = "Track 1. Ready" ;
	CMainFrame *pFrame = (CMainFrame*)AfxGetApp()->m_pMainWnd;
	CPlayerView *pView = (CPlayerView*) pFrame->GetActiveView();
	pView->Invalidate();
	m_s.StopSoundFile();

}


void CPlayer::z2()
{
	CString strNum;
	strNum.Format("%d", nTrackNumber);
```

```
	m_strTitle = "Track " + strNum + ". Playing" ;
	CMainFrame *pFrame = (CMainFrame*)AfxGetApp()->m_pMainWnd;
	CPlayerView *pView = (CPlayerView*) pFrame->GetActiveView();
	pView->Invalidate();
	m_s.PlaySoundFile(const_cast<char*>(m_strFile.c_str()));
}

void CPlayer::z3()
{
	if (nTrackNumber < 7) {
		nTrackNumber++;
		CString str;
		str.Format("%d", nTrackNumber);
		m_strFile = "mus/" + str + ".wav";
	}
	A0(0); // безусловный переход

}

void CPlayer::z4()
{
	if (nTrackNumber > 0) {
		if (nTrackNumber > 1)
		{ nTrackNumber--; }
		CString str;
		str.Format("%d", nTrackNumber);
		m_strFile = "mus/" + str + ".wav";
	}

	A0(0);
}

void CPlayer::z5()
{
	CString strNum;
	strNum.Format("%d", nTrackNumber);
	m_strTitle = "Track " + strNum + ". Paused" ;

	CMainFrame *pFrame = (CMainFrame*)AfxGetApp()->m_pMainWnd;
	CPlayerView *pView = (CPlayerView*) pFrame->GetActiveView();
	m_s.StopSoundFile();
	pView->Invalidate();
}

void CPlayer::z6()
{
	m_strTitle = "NO TRACK";

	CMainFrame *pFrame = (CMainFrame*)AfxGetApp()->m_pMainWnd;
	CPlayerView *pView = (CPlayerView*) pFrame->GetActiveView();
	pView->Invalidate();
}


void CPlayer::BtnPlayPressed()
{
	A0(1);
```

```
}

void CPlayer::BtnStopPressed()
{
	A0(2);
}

void CPlayer::BtnNextPressed()
{
	A0(3);
}

void CPlayer::BtnPrevPressed()
{
	A0(4);
}

void CPlayer::BtnPausePressed()
{
	A0(5);
}

void CPlayer::FindNoTracks()
{
	A0(6);
}

BEGIN_MESSAGE_MAP(CPlayer, CWnd)
//{{AFX_MSG_MAP(CPlayer)
ON_WM_TIMER()
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

void CPlayer::OnTimer(UINT nIDEvent)
{
	if(nIDEvent == 1000)
	{
	int i = 0;
	};
}
```

## 2. Фрагменты исходного кода класса *CPlayerView*

```
void CPlayerView::OnSize(UINT nType, int cx, int cy)
{
	CView::OnSize(nType, cx, cy);
	CPlayer::Instance()->SetNewSize(CSize(cx, cy));
}

void CPlayerView::OnFinalRelease()
{
	CView::OnFinalRelease();
}

BOOL CPlayerView::OnEraseBkgnd(CDC* pDC)
{
```

```
        return true;
        return CView::OnEraseBkgnd(pDC);
}

// Обработка действий пользователя
void CPlayerView::OnLButtonDown(UINT nFlags, CPoint point)
{
        switch(CheckMouseDown(point)) {
        case 0:
                CPlayer::Instance()->BtnPlayPressed();
                break;
        case 1:
                CPlayer::Instance()->BtnStopPressed();
                break;
        case 2:
                CPlayer::Instance()->BtnPausePressed();
                break;
        case 3:
                CPlayer::Instance()->BtnPrevPressed();
                break;
        case 4:
                CPlayer::Instance()->BtnNextPressed();
                break;
        }

        CView::OnLButtonDown(nFlags, point);
}

int CPlayerView::CheckMouseDown(CPoint pt)
{
        CSize size = CPlayer::Instance()->GetSize();
        if ((pt.y > (size.cy - size.cy/8))&& pt.y < size.cy)
                for (int i = 0; i < 5; ++i)
                {
                        if ((i*size.cx/5 < pt.x) && (pt.x < (i+1)*size.cx/5))
                                return i;
                }
        return -1;
}
```

# Заключение

Применение автоматного подхода в этой задаче позволило эффективно сконцентрировать логику программы в одном классе.

# Литература


1. www.microsoft.com
2. www.xmms.org
3. *Шалыто А.А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
4. is.ifmo.ru