타이머

MFC 윈도우 프로그래밍

1. 타이머 메시지

- 컴퓨터에 동작하는 시계를 이용하여 일정 단위 시간마다 신호를 받을 수 있다.
- 이렇게 주기적으로 받는 신호가 타이머 메시지이다.

타어머 설정 함수

```
UINT_PTR CWnd::SetTimer(
   UINT_PTR nIDEvent,
   UINT nElapse,
   void (CALLBACK* lpfnTimer)(HWND,UINT,UINT_PTR,DWORD)
);
```

- nIDEvent: 정수이며 타이머의 ID이다. 타이머가 여러 개 존재할 수 있으므로 이를 구분하는 데 사용한다.
- nElapse: 타이머 메시지를 보낼 시간 간격으로 단위는 밀리초이다. 예를 들어 1000은 1초를 의미한다.
- lpfnTimer: 타이머가 보내는 WM_TIMER 메시지를 받을 함수 이름이다. NULL이면 같은 클래스에 추가된 WM_TIMER 이벤트 처리 함수인 OnTimer()가 메시지를 받는다. 타이머에 따라 처리 함수를 달리하려면 다음과 같은 함수를 만들고 함수 이름을 쓴다.

```
void CALLBACK TimerProc(
   HWND hWnd,              // 타이머를 설정한 윈도우
   UINT nMsg,              // 타이머 메시지
   UINT_PTR nIDEvent,      // 타이머의 ID
   DWORD dwTime            // 시스템 시간
);
```

1. 타이머 메시지

타이머 설정의 예

```
void CProjectView::OnLButtonDown()
{
    SetTimer(1, 700, NULL);
    SetTimer(2, 1000, &CProjectView::TimerProc);
    CView::OnLButtonDown();
}

void CProjectView::OnTimer(UINT nIDEvent)
{
    // 0.7초마다 수행
    CView::OnTimer(nIDEvent);
}

void CALLBACK CProjectView::TimerProc(HWND hWnd, UINT nMsg,
                                      UINT_PTR nIDEvent,DWORD dwTime)
{
    // 1초마다 수행
}
```

1. 타이머 메시지

타이머 정지 함수

```
BOOL CWnd::KillTimer(
    UINT_PTR nIDEvent,

);
```

- nIDEvent: 정수이며 동작 중인 타이머의 ID이다.

1. 타이머 메시지

4-1 자동으로 원 이동하기

발생하는 이벤트	이벤트 처리 함수	해야 할 일
WM_LBUTTONDOWN	OnLButtonDown( )	• 타이머를 동작시킴 • 마우스 좌표와 원의 중심 좌표를 가지고 타이머 신호에 따라 이동할 단위 거리를 구함
WM_TIMER	OnTimer( )	• 원의 중심 좌표에 단위 거리를 더함
	OnDraw( )	• 원을 중심 좌표에 출력함

• 클래스에 멤버변수 추가

클래스	액세스 지정자	자료형	변수 이름
CProjectView	public	int	m_Count
CProjectView	public	CPoint	m_ptCircle
CProjectView	public	int	m_xStep
CProjectView	public	int	m_yStep

1. 타이머 메시지

• 이벤트 처리 함수 추가

클래스	이벤트	처리 함수
CProjectView	WM_TIMER	OnTimer( )
CProjectView	WM_LBUTTONDOWN	OnLButtonDown( )

• 함수의 코드 수정

```
CProjectView::CProjectView()
: m_ptCircle(0)
, m_xStep(0)
, m_yStep(0)
, m_Count(0)
{
      m_ptCircle = CPoint(100,100);
}
```

1. 타이머 메시지

```
01  void CProjectView::OnLButtonDown(UINT nFlags, CPoint point)
02  {
03          m_xStep = (point.x - m_ptCircle.x)/10;
04          m_yStep = (point.y - m_ptCircle.y)/10;
05          m_Count = 0;
06          SetTimer(1, 100, NULL);
07          CView::OnLButtonDown(nFlags, point);
08  }
```

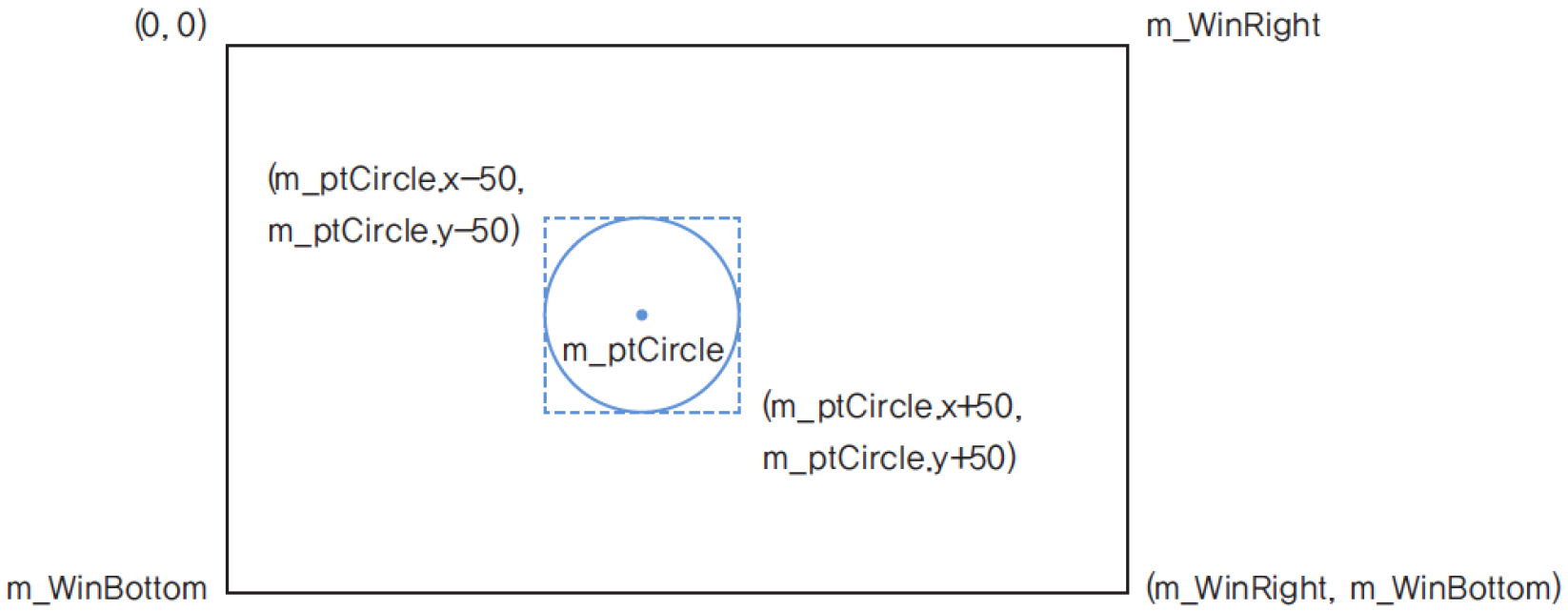
```
01  void CProjectView::OnTimer(UINT_PTR nIDEvent)
02  {
03          m_Count = m_Count + 1;
04          m_ptCircle.x += m_xStep;
05          m_ptCircle.y += m_yStep;
06          if (m_Count == 10)
07                  KillTimer(1);
08          Invalidate();
09          CView::OnTimer(nIDEvent);
10  }
```

1. 타이머 메시지

```
void CProjectView::OnDraw(CDC* pDC)
{
	pDC->Ellipse(m_ptCircle.x-50, m_ptCircle.y-50,
			m_ptCircle.x+50, m_ptCircle.y+50);
}
```

2. 윈도우 뷰 영역 내 객체의 바운드

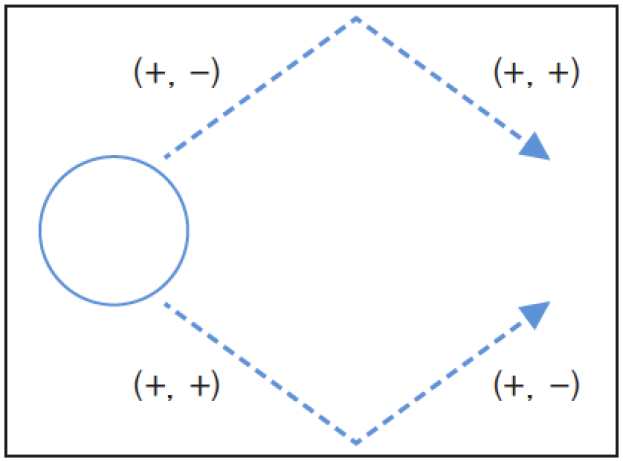
뷰 영역과 원의 경계 좌표

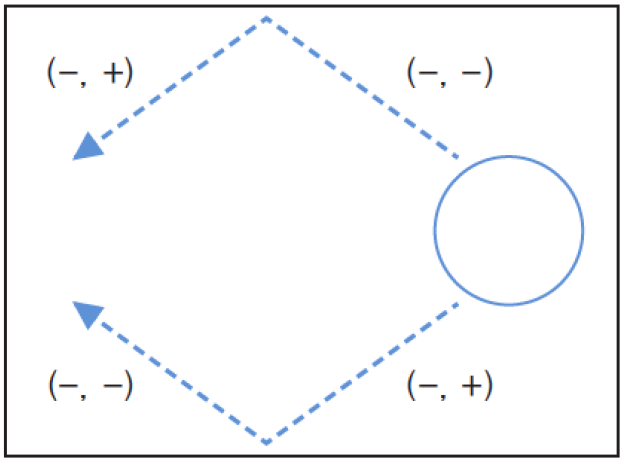


2. 윈도우 뷰 영역 내 객체의 바운드

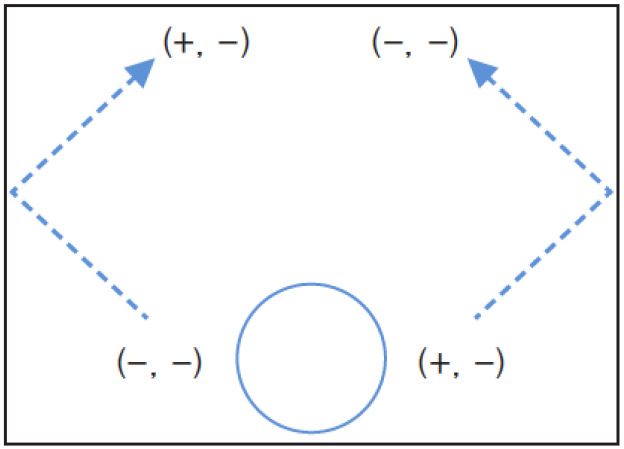
- 바운드 처리

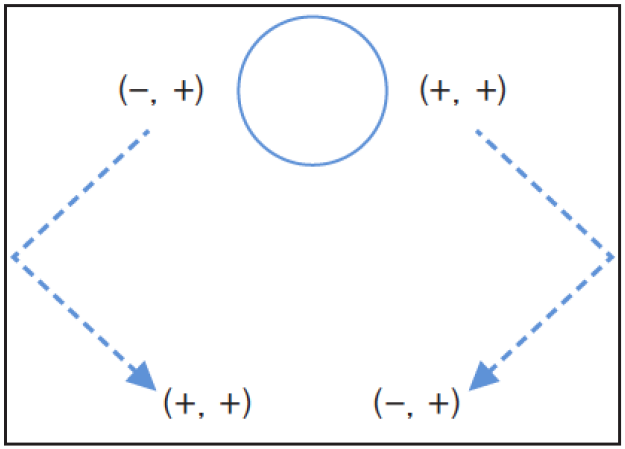
뷰 영역의 위 또는 아래 경계 충돌 후 원의 진행 방향





뷰 영역의 왼쪽 또는 오른쪽 경계 충돌 후 원의 진행 방향





2. 윈도우 뷰 영역 내 객체의 바운드

뷰 영역 내에서 원을 바운드시키기

발생하는 이벤트	이벤트 처리 함수	해야 할 일
WM_LBUTTONDOWN	OnLButtonDown( )	• 타이머를 동작시킴 • 마우스 좌표와 원의 중심 좌표를 가지고 타이머 신호에 따라 이동할 단위 거리를 구함
WM_TIMER	OnTimer( )	• 원의 중심 좌표에 단위 거리를 더함
WM_SIZE	OnSize( )	• 뷰 영역의 오른쪽 경계와 하단 경계를 저장함
	OnDraw( )	• 원을 중심 좌표에 출력함

- **클래스에 멤버변수 추가**

클래스	액세스 지정자	자료형	변수 이름
CProjectView	public	int	m_WinRight
CProjectView	public	int	m_WinBottom
CProjectView	public	CPoint	m_ptCircle
CProjectView	public	int	m_xStep
CProjectView	public	int	m_yStep

2. 윈도우 뷰 영역 내 객체의 바운드

• 이벤트 처리 함수 추가

클래스	이벤트	처리 함수
CProjectView	WM_TIMER	OnTimer( )
CProjectView	WM_LBUTTONDOWN	OnLButtonDown( )
CProjectView	WM_SIZE	OnSize( )

• 함수의 코드 수정

```
CProjectView::CProjectView()
: m_ptCircle(0)
, m_xStep(0)
, m_yStep(0)
, m_WinBottom(0)
, m_WinRight(0)
{
	m_ptCircle = CPoint(100,100);
}
```

2. 윈도우 뷰 영역 내 객체의 바운드

```
01  void CProjectView::OnLButtonDown(UINT nFlags, CPoint point)
02  {
03          m_xStep = (point.x - m_ptCircle.x)/10;
04          m_yStep = (point.y - m_ptCircle.y)/10;
05          SetTimer(1, 100, NULL);
06          CView::OnLButtonDown(nFlags, point);
07  }
```

2. 윈도우 뷰 영역 내 객체의 바운드

```
01  void CProjectView::OnTimer(UINT_PTR nIDEvent)
02  {
03        m_ptCircle.x += m_xStep;
04        m_ptCircle.y += m_yStep;
05        if (m_ptCircle.x - 50 < 0 || m_ptCircle.x + 50 > m_WinRight)
06        {
07              m_ptCircle.x -= m_xStep;
08              m_xStep *= -1;
09        }
10        if (m_ptCircle.y - 50 < 0 || m_ptCircle.y + 50> m_WinBottom)
11        {
12              m_ptCircle.y -= m_yStep;
13              m_yStep *= -1;
14        }
15        Invalidate();
16        CView::OnTimer(nIDEvent);
17  }
```

2. 윈도우 뷰 영역 내 객체의 바운드

```
void CProjectView::OnSize(UINT nType, int cx, int cy)
{
	CView::OnSize(nType, cx, cy);
	m_WinRight = cx;
	m_WinBottom = cy;
}
```

```
void CProjectView::OnDraw(CDC* pDC)
{
	pDC->Ellipse(m_ptCircle.x-50, m_ptCircle.y-50,
			m_ptCircle.x+50, m_ptCircle.y+50);
}
```

Thank You !

MFC 윈도우 프로그래밍