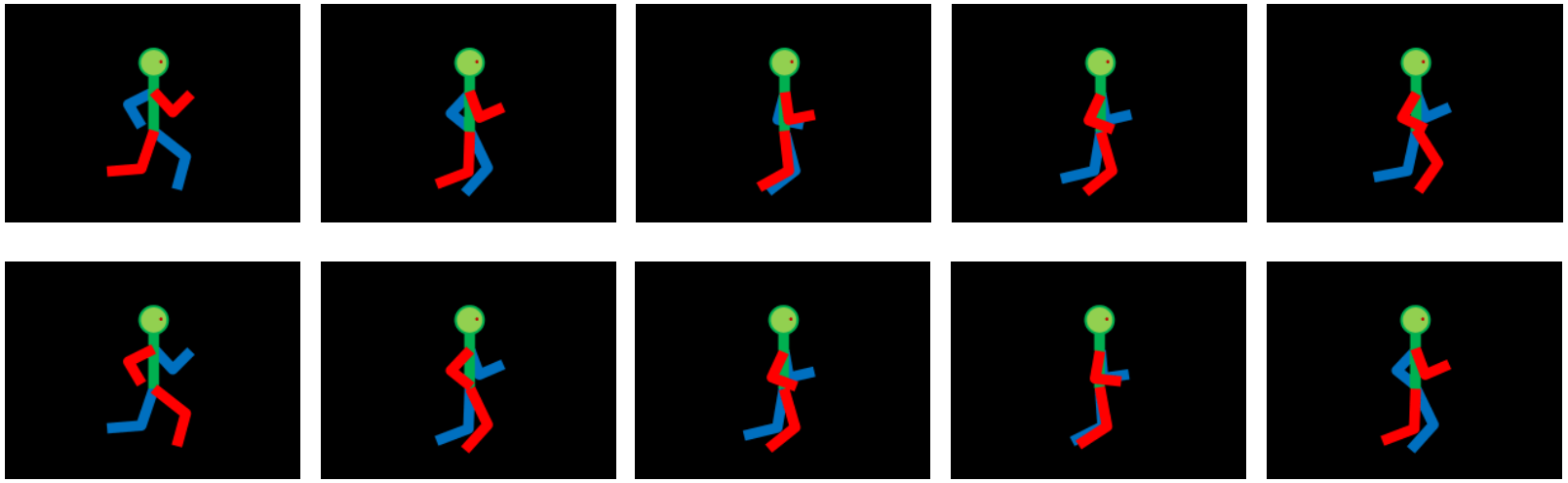
제 5장 비트맵과 애니메이션 2

2023년 1학기 윈도우 프로그래밍

3. 비트맵 애니메이션

- **애니메이션**
  - 각 시점에 다른 그림을 그려서 움직이는 효과를 얻는다.
    - 프레임(Frame): 움직이는 그림 중 하나의 동작이 그려진 이미지
  - 애니메이션 동작은 **타이머로** 처리한다.
  - 매 타이머의 주기에 각 프레임을 표시하여, 각 동작에 하나의 프레임 만을 보여준다.
  - 애니메이션 이미지들이 한 파일에 저장되어 있을 때는 한 프레임씩 이동하면서 필요한 부분을 잘라내어 번갈아 표시한다. 오프셋 개념을 이용한다
    - 오프셋 (Offset): 동일 오브젝트 안에서 오브젝트 처음부터 주어진 요소나 지점까지의 변위차를 나타내는 정수형



비트맵 애니메이션

- 사용 예) 10개의 이미지를 사용하여 애니메이션 구현

```
HBITMAP RunBit[10];
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
        static int xPos=0;

        switch (iMsg)
        {
        case WM_CREATE:
                RunBit[0]= LoadBitmap (hInst, MAKEINTRESOURCE(IDB_BITMAP_R1));          //-- 필요한 10개의 애니메이션 이미지들을 로드하기
                RunBit[1]= LoadBitmap (hInst, MAKEINTRESOURCE(IDB_BITMAP_R2));
                RunBit[2]= LoadBitmap (hInst, MAKEINTRESOURCE(IDB_BITMAP_R3));
                . . . . . .
                RunBit[9]= LoadBitmap (hInst, MAKEINTRESOURCE(IDB_BITMAP_R10));
                SetTimer(hwnd, 1, 100, NULL);
                break;
        case WM_TIMER:
                xPos += 10;                                                             //--- 애니메이션의 x 위치 변경하기
                if (xPos > 800)                 xPos = 0;
                InvalidateRect (hwnd, NULL, true);
                return 0;
        case WM_PAINT:
                hdc = BeginPaint(hwnd, &ps);
                Animation (xPos, 300, hdc);                                             //--- (xPos, 300) 위치에 애니메이션 그리기
                EndPaint(hwnd, &ps);
                break;
        case WM_DESTROY:
                for (int i = 0; i < 10; i++)            DeleteObject (RunBit[i]);
                PostQuitMessage (0);
        }
        return DefWindowProc (hwnd, iMsg, wParam, lParam);
}
```

비트맵 애니메이션

//--- 10개의 이미지를 사용하여 애니메이션 구현

```
void Animation (int xPos, int yPos, HDC hdc)
{
	HDC    memdc;
	HBITMAP hBit, oldBit;
	static int count;
	int i;

	count++;
	count = count % 10;

	hBit = LoadBitmap (hInst, MAKEINTRESOURCE(IDB_BITMAP_BACK));	//--- 배경 이미지 로드하기

	memdc = CreateCompatibleDC (hdc);

	oldBit = (HBITMAP) SelectObject (memdc, hBit);			//--- 배경 이미지를 메모리 DC에 올리기
	BitBlt (hdc, 0, 0, 800, 600, memdc, 0, 0, SRCCOPY);

	(HBITMAP) SelectObject (memdc, RunBit[count]);			//--- 순서대로 전경 이미지를 메모리 DC에 올리기
	BitBlt (hdc, xPos, yPos, 64, 64, memdc, 0, 0, SRCCOPY);

	SelectObject (memdc, oldBit);

	DeleteDC (memdc);						//--- 메모리 DC와 로드한 배경 이미지 삭제하기
	DeleteObject (hBit);
}
```

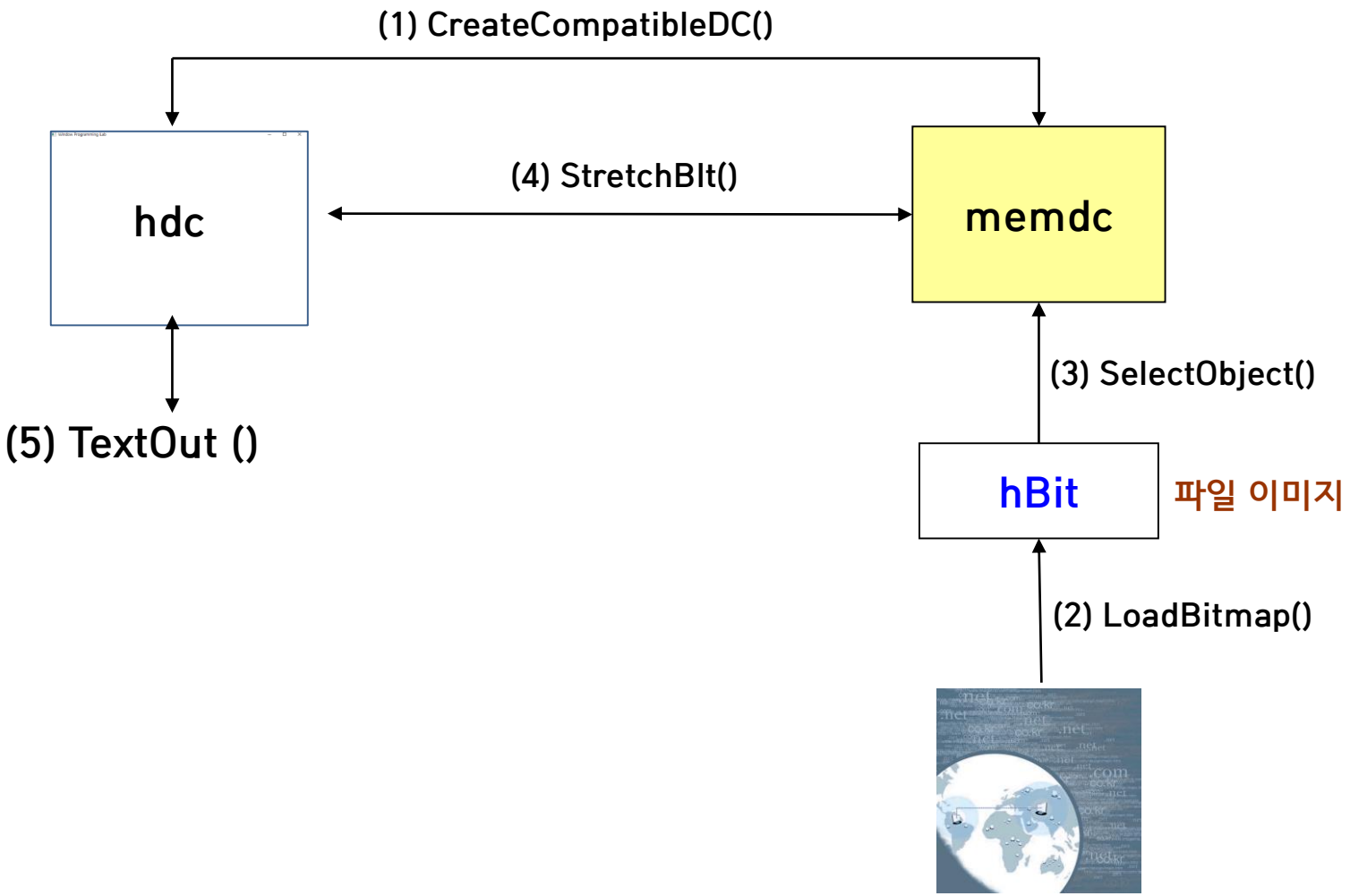
4. 더블 버퍼링

- **비트맵 이미지 여러 개를 이용하여 애니메이션을 나타낼 때**
  - 이미지를 순서대로 화면에 출력
  - 예를 들어 풍경 위에 달리는 강아지를 표현한다면
    1. 풍경 이미지를 먼저 출력
    2. 그 다음에 강아지 이미지를 출력
    3. 달리는 모습을 나타내고자 한다면 풍경 이미지 출력과 강아지 이미지 출력을 번갈아 가며 계속 수행
  - 이미지의 잦은 출력으로 인해 화면이 자주 깜박거리는 문제점

- **문제점 해결**
  - 메모리 디바이스 컨텍스트를 하나 더 사용
  - 추가된 메모리 디바이스 컨텍스트에 그리기를 원하는 그림들을 모두 출력한 다음 화면 디바이스 컨텍스트로 한꺼번에 옮기는 방법을 이용

- **추가된 메모리 디바이스 컨텍스트가 추가된 버퍼 역할을 하기 때문에 이 방법을 더블버퍼링이라 부름**

기존 방법: 배경화면 위로 움직이는 글



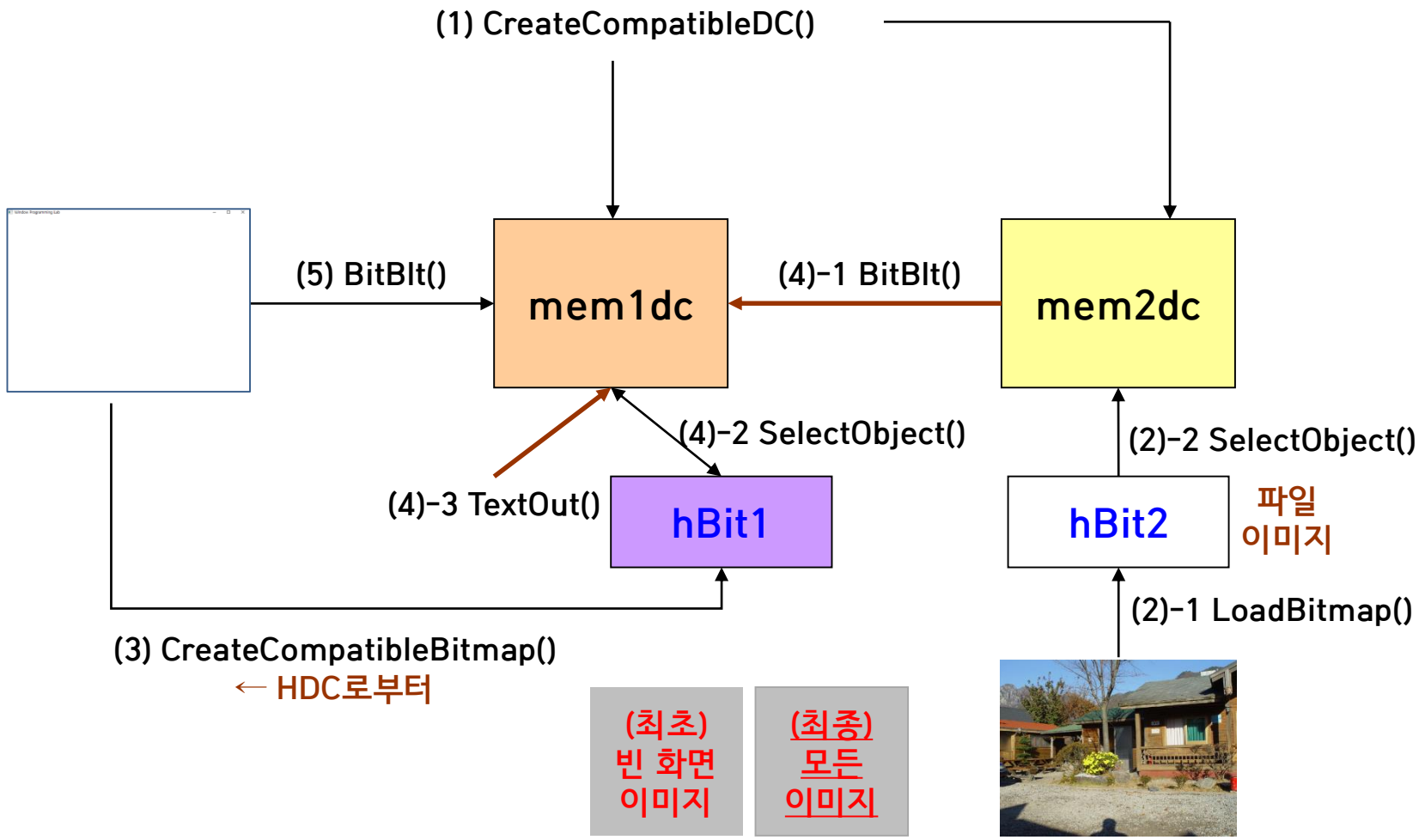
기존 방법: 배경화면 위로 움직이는 글

```
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
        HDC hdc, memdc;
        static HBITMAP hBit, oldBit;
        TCHAR word[] = L"움직이는 그림";

        switch(iMsg) {

                case WM_CREATE:
                    yPos = -30;                                    //--- -30: 글자의 높이 고려
                    GetClientRect (hwnd, &rectView);
                    SetTimer (hwnd, 1, 70, NULL);
                    hBit=LoadBitmap (hInstance, MAKEINTRESOURCE(IDB_BITMAP1));
                    break;

                case WM_TIMER:                                     //--- Timeout 마다 y좌표 변경 후, 출력 요청
                    yPos += 5;
                    if (yPos > rectView.bottom)
                                yPos = -30;
                    InvalidateRect (hwnd, NULL, true);
                    break;
```

기존 방법: 배경화면 위로 움직이는 글

```
            case WM_PAINT:
                hdc=BeginPaint(hwnd, &ps);

            //--- 이미지 로드
                hBit=LoadBitmap (hInstance, MAKEINTRESOURCE(IDB_BITMAP1));
                memdc = CreateCompatibleDC (hdc);

            //--- 이미지 출력
                oldBit=(HBITMAP)SelectObject(memdc, hBit);

            //--- 메모리 DC → 화면 DC(hdc)로 이동, 출력
                StretchBlt (hdc, 0, 0, rectView.right, rectView.bottom,          memdc, 0, 0, rectView.right, rectView.bottom, SRCCOPY);
                SelectObject (memdc, oldBit);
                DeleteDC (memdc);

            //--- 문자열 출력
                TextOut(hdc, 200, yPos, word, strlen(word));

                EndPaint(hwnd, &ps);
                break;

        }
        return DefWindowProc (hwnd, iMsg, wParam, lParam)
}
```

더블 버퍼링 적용 예 1) 배경화면 위로 움직이는 글



더블 버퍼링 적용 예 1) 배경화면 위로 움직이는 글

```
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{

    HDC hdc, memdc;
    HDC mem1dc, mem2dc;
    static HBITMAP hBit1, hBit2;
    HBITMAP oldBit1, oldBit2;
    ...
    TCHAR word[] = L"더블 버퍼링 실습";


    switch(iMsg) {
        case WM_CREATE:
            yPos = -30;
            GetClientRect(hwnd, &rectView);
            SetTimer(hwnd, 1, 70, NULL);

            //--- hBit2에 배경 그림 로드, 나중에 mem2dc에 hBit2 그림 설정
            hBit2 = LoadBitmap ( hInstance, MAKEINTRESOURCE(IDB_BITMAP4));
            break;
```

더블 버퍼링 적용 예 1) 배경화면 위로 움직이는 글

```
case WM_TIMER:
        yPos += 5;
        if (yPos > rectView.bottom)              yPos = -30;
        hdc = GetDC(hwnd);

        if (hBit1 == NULL)                                          //--- hBit1을 hdc와 호환되게 만들어준다.
        hBit1 = CreateCompatibleBitmap (hdc, 1024, 768);

        //--- hdc와 호환되는 mem1dc를 만들어준다.
        mem1dc = CreateCompatibleDC (hdc);

        //---  mem1dc와 호환되는 mem2dc를 만들어준다.
        mem2dc = CreateCompatibleDC (mem1dc);

        //--- mem2dc의 비트맵을 mem1dc에 옮기고, mem1dc를 hdc로 옮기려고 함
        //--- hBit1의 이미지를 mem1dc로, hBit2의 이미지를 mem2dc로 선택
        oldBit1 = (HBITMAP) SelectObject (mem1dc, hBit1);          //--- mem1dc에는 hBit1
        oldBit2 = (HBITMAP) SelectObject (mem2dc, hBit2);          //--- mem2dc에는 hBit2: hBit2에는 배경 그림이 저장되어 있음

        //--- mem2dc에 있는 배경그림을 mem1dc에 옮긴다.
        BitBlt(mem1dc, 0, 0, 1024, 768, mem2dc, 0, 0, SRCCOPY);

        SetBkMode (mem1dc, TRANSPARENT);
        TextPrint (mem1dc, 200, yPos, word);                       //--- mem1dc에 텍스트 출력

        //--- 저장한 비트맵 핸들값을 DC에 원상복귀, 생성된 MDC 삭제
        SelectObject (mem2dc, oldBit2);                  DeleteDC (mem2dc);
        SelectObject (mem1dc, oldBit1);                  DeleteDC (mem1dc);
        ReleaseDC (hwnd, hdc);
        InvalidateRgn (hwnd, NULL, false);          //--- 다시 그리기를 할 때 배경 이미지를 지우지 않고 출력하기 위해 마지막 인자를 false로 설정
        break;
```

더블 버퍼링 적용 예 1) 배경화면 위로 움직이는 글

```
    case WM_PAINT:
            GetClientRect (hwnd, &rectView);
            hdc = BeginPaint (hwnd, &ps);

            mem1dc = CreateCompatibleDC (hdc);

            //--- hBit1에는 배경과 텍스트가 출력된 비트맵이 저장되어 있다.  이 비트맵을 mem1dc에 선택
            oldBit1 = (HBITMAP) SelectObject (mem1dc, hBit1);

            //--- mem1dc에 있는 내용을 hdc에 복사한다.
            BitBlt (hdc, 0, 0, 1024, 768, mem1dc, 0, 0, SRCCOPY);

            SelectObject (mem1dc, oldBit1);
            DeleteDC (mem2dc);
            EndPaint (hwnd, &ps);
            break;
    }
    return DefWindowProc (hwnd, iMsg, wParam, lParam)
}
```

더블 버퍼링을 위해 비트맵과 DC 생성 함수

- **사용 함수**

HDC CreateCompatibleDC (HDC hdc);

- 주어진 DC와 호환되는 메모리 DC를 생성해 준다.
- 주어진 DC가 사용하는 출력장치의 종류나 출력장치가 사용중인 그래픽 드라이버 정보를 가지고 새로운 DC 를 만든다.
- hdc와 동일한 방법으로 그림을 그리지만 화면에 출력은 되지 않는다.

  - HDC hdc: 주어진 DC

HBITMAP CreateCompatibleBitmap (HDC hdc, int nWidth, int nHeight);

- hdc와 호환되는 비트맵을 생성하여 반환하는 함수
- 화면 DC와 호환되게 만들어야 한다. (메모리 DC와 호환되게 만들면 1비트 색상수를 사용하는 비트맵을 생성하게 된다.)
- CreateCompatibleDC로 생성한 DC를 사용하려면 비트맵 객체를 만들어서 연결하여 사용
- 생성된 비트맵은 hdc와 호환되는 어떤 메모리 DC에서도 선택되어질 수 있다.

  - HDC hdc: DC 핸들
  - int nWidth/nHeight: 작성하는 비트맵의 가로/세로 사이즈

더블 버퍼링 적용 예 2) 배경화면 위로 움직이는 애니메이션

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT iMessage, WPARAM wParam, LPARAM lParam)
{
	HDC hdc;
	HDC MemDC, MemDCImage;
	HBITMAP OldBit[3];

	switch (iMessage) {
	case WM_CREATE:
		hdc = GetDC(hWnd);

		hBitBackground = LoadBitmap (g_hInst, MAKEINTRESOURCE(IDB_BITMAP1));		//--- 배경 이미지 1개
		hBitCha[0] = LoadBitmap (g_hInst, MAKEINTRESOURCE(IDB_BITMAP2));		//---- 애니메이션 이미지 6개
		hBitCha[1] = LoadBitmap (g_hInst, MAKEINTRESOURCE(IDB_BITMAP3));
		hBitCha[2] = LoadBitmap (g_hInst, MAKEINTRESOURCE(IDB_BITMAP4));
		hBitCha[3] = LoadBitmap (g_hInst, MAKEINTRESOURCE(IDB_BITMAP5));
		hBitCha[4] = LoadBitmap (g_hInst, MAKEINTRESOURCE(IDB_BITMAP6));
		hBitCha[5] = LoadBitmap (g_hInst, MAKEINTRESOURCE(IDB_BITMAP7));
		hBit = CreateCompatibleBitmap (hdc, WindowWidth, WindowHeight);		//--- 이미지들을 저장할 비트맵 생성: 전역변수 또는 static 타입으로 선언
	break;

	case WM_PAINT:
		hdc = BeginPaint(hWnd, &ps);

		//--- 더블 버퍼링 사용
		MemDC = CreateCompatibleDC (hdc);
		OldBit[0] = (HBITMAP) SelectObject (MemDC, hBit);		//--- hBit: 이미지가 저장되어 있는 비트맵 전역 변수
		BitBlt (hdc, 0, 0, WindowWidth, WindowHeight, MemDC, 0, 0, SRCCOPY);
		SelectObject (MemDC, OldBit[0]);
		DeleteDC (MemDC);
		EndPaint (hWnd, &ps);
		break;
```

배경 이미지 위에 애니메이션이 그려진다.
더블 버퍼링을 사용한다.
일부 변수는 선언이 빠져 있음.
확인하여 전역 또는 지역 변수로 추가.

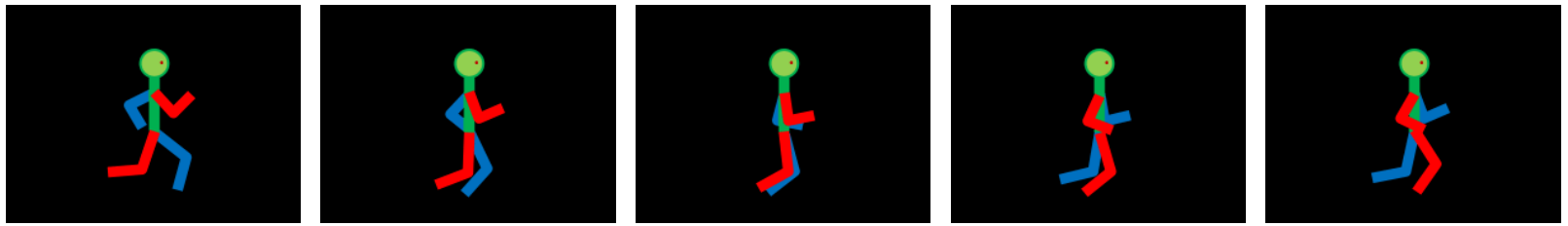
더블 버퍼링 적용 예 2) 배경화면 위로 움직이는 애니메이션

```
case WM_TIMER:
	hdc = GetDC(hWnd);

	chaX += 5;
	if (chaX > WindowWidth)                                          //--- 전역변수로 선언한 애니메이션 x 좌표
		chaX = 0;

	//--- 더블 버퍼링 사용
	MemDC = CreateCompatibleDC (hdc);                                //--- 더블 버퍼로 사용 할 메모리 DC 생성
	MemDCImage = CreateCompatibleDC (MemDC);                         //--- 이미지를 저장 할  메모리 DC 생성

	OldBit[0] = (HBITMAP) SelectObject (MemDC, hBit);                //--- MemDC와 hBit 를 호환되게 선택함

	OldBit[1] = (HBITMAP) SelectObject (MemDCImage, hBitBackground); //--- 배경 이미지
	BitBlt (MemDC, 0, 0, WindowWidth, WindowHeight, MemDCImage, 0, 0, SRCCOPY);

	OldBit[2] = (HBITMAP) SelectObject (MemDCImage, hBitCha[chaCount]); //--- 애니메이션 이미지
	StretchBlt (MemDC, chaX, chaY, chaWidth, chaHeight, MemDCImage, 0, 0, chaWidth, chaHeight, SRCCOPY);
                                                                     //--- MemDC에 배경과 애니메이션 저장 → hBit 비트맵에 저장

	chaCount++;                          //--- 애니메이션 이미지 순서
	chaCount %= 6;

	DeleteDC (MemDC);
	DeleteObject (MemDCImage);
	ReleaseDC (hWnd, hdc);

	InvalidateRect (hWnd, NULL, false);                              //--- 그리기를 위해 InvalidateRect 함수 호출 (마지막 변수는 false)
break;
```

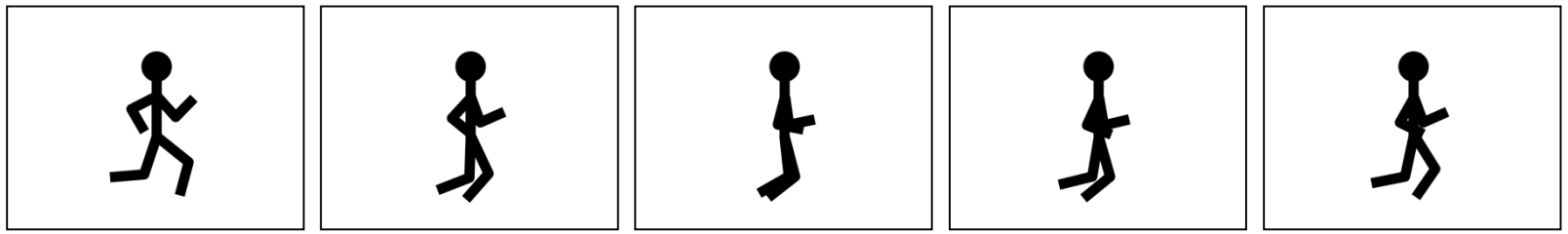
5. 비트맵 마스크

- 사각형의 비트맵 이미지에서 원하는 부분만을 사용하고 싶을 때, 그리려는 비트맵 이미지 부분에 마스크를 씌운다.

  - 필요한 이미지:
    - 비트맵 이미지
    - 출력하고자 하는 부분을 흑색 처리한 마스크



비트맵 이미지



마스크 이미지

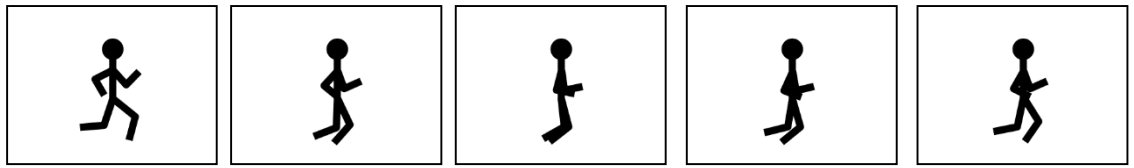
비트맵 마스크

- **처리 방법:**
  - 각 프레임의 동작마다 마스크 이미지와 출력하고자 하는 소스 이미지의 그림을 각각 두번씩 씌워주어야 한다.

1. 소스의 원하는 부분을 흑백으로 처리한 마스크를 배경 그림과 **AND** 연산 → 배경 이미지에 흑색 마스크가 그려진다.

   BitBlt (hdc, x, y, size_x, size_y, BitmapMaskDC, mem_x, mem_y, **SRCAND**);
   - SRCAND: 소스와 대상의 AND 연산값으로 칠한다.
     - » 마스크와 배경이미지의 AND 연산

배경 이미지

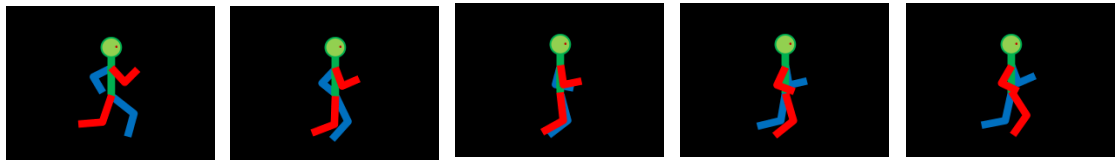


마스크 이미지

2. 여기에 원하는 그림을 배경 그림과 OR 연산 → 배경과 합성된 이미지로 나타나게 된다.

   BitBlt (hdc, x, y, size_x, size_y, hBitmapFrontDC, mem_x, mem_y, **SRCPAINT**);
   - SRCPAINT: 소스와 대상의 OR 연산값으로 칠한다.
     - » 출력하고자 하는 이미지와 배경이미지의 OR 연산

배경 이미지



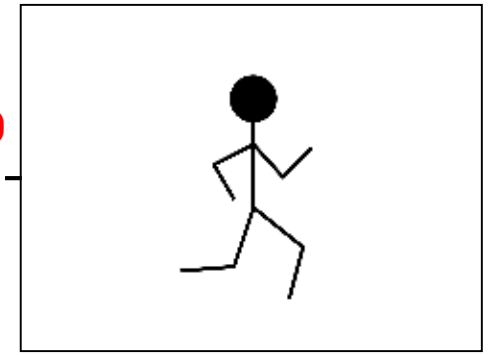
비트맵 이미지

비트맵 마스크

1. 마스크 그리기: AND 연산

AND

흰색(255,255,255)



컬러(X, X, X)

검은색(0, 0, 0)

흰색(255,255,255)
AND 컬러(X , X , X)

컬러(X , X , X)

검은색(0, 0, 0)
AND 컬러 (X, X, X)

검은색(0, 0, 0)

비트맵 마스크

2. 캐릭터 그리기: OR 연산

컬러(X, X, X)
검은색(0, 0, 0)

OR

검은색(0, 0, 0)

컬러(X, X, X)

		검은색(0,	0,	0)
OR		컬러 (X,	X,	X)
		컬러 (X,	X,	X)

비트맵 마스크

3. 결과: 배경 위에 캐릭터가 올려진 결과 이미지

비트맵 마스크

```
HBITMAP       RunBit[5], Mask[5];

LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
        HDC hdc;
        PAINTSTRUCT ps;
        static int xPos=0, yPos;

        switch (iMsg)
        {
        case WM_CREATE:
                RunBit[0]= LoadBitmap(hInst, MAKEINTRESOURCE(IDB_BITMAP_R1));     //--- 애니메이션 이미지 로드하기: 5개
                RunBit[4]= LoadBitmap(hInst, MAKEINTRESOURCE(IDB_BITMAP_R5));

                Mask[0] = LoadBitmap(hInst, MAKEINTRESOURCE(IDB_BITMAP_M1));     //--- 마스크 이미지 로드하기: 5개
                Mask[4] = LoadBitmap(hInst, MAKEINTRESOURCE(IDB_BITMAP_M5));
                SetTimer (hwnd, 1000, 100, NULL);
                break;
        case WM_PAINT:
                hdc = BeginPaint (hwnd, &ps);
                Animation (xPos, yPos, hdc);
                EndPaint (hwnd, hdc);
                break;
        case WM_TIMER:
                xPos += 10;
                InvalidateRect (hwnd, NULL, false);
                break;
        case WM_DESTROY:
                for (i = 0; i < 10; i++) {
                     DeleteObject (RunBit[i]);      DeleteObject (Mask[i]);
                }
                PostQuitMessage (0);
                break;
        }
        return DefWindowProc (hWnd, iMessage, wParam, lParam);
}
```

비트맵 마스크

```
void Animation (int xPos, int yPos, HDC hdc)
{
        HDC     memdc;
        static int count;
        int       i;

        count++;

        memdc = CreateCompatibleDC(hdc);
        hBit = LoadBitmap(hInst, MAKEINTRESOURCE(IDB_BITMAP5));     // 배경 이미지
        (HBITMAP)SelectObject(memdc, hBit);
        BitBlt(hdc, 0, 0, 819, 614, memdc, 0, 0, SRCCOPY);

        (HBITMAP) SelectObject (memdc, Mask[count]);
        BitBlt (hdc, xPos, yPos, 180, 240, memdc, 0, 0, SRCAND);

        (HBITMAP) SelectObject (memdc, RunBit[count]);
        BitBlt (hdc, xPos, yPos, 180, 240, memdc, 0, 0, SRCPAINT);

        DeleteDC (memdc);
        DeleteObject (hBit);
}
```

마스크를 그린다.

캐릭터를 그린다.

6. 투명 비트맵 처리

- **비트맵의 일부를 투명하게 처리하여 투명색 부분은 출력에서 제외한다.**
  - 1개의 특정 색을 투명하게 설정한다.
  - BitBlt 함수나 StretchBlt 함수 대신 사용할 수 있다.

BOOL TransparentBlt (HDC hdcDest, int nXOriginDest, int nYOriginDest, int nWidthDest, int hHeightDest,
HDC hdcSrc, int nXOriginSrc, int nYOriginSrc, int nWidthSrc, int nHeightSrc, UINT crTransparent);

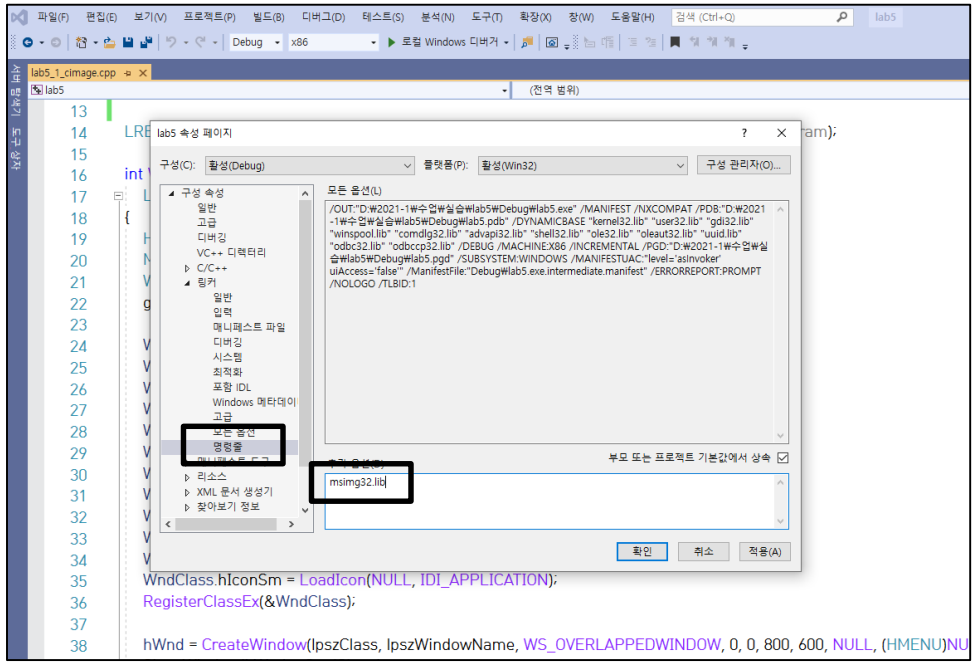
- 비트맵의 특정 색을 투명하게 처리하는 함수
  - HDC hdcDest: 출력할 목표 DC 핸들
  - int nXOriginDest : 좌측 상단의 x 좌표값
  - int nYOriginDest : 좌측 상단의 y 좌표값
  - int nWidthDest : 목표 사각형의 넓이
  - int hHeightDest : 목표 사각형의 높이

  - HDC hdcSrc : 소스 DC 핸들
  - int nXOriginSrc : 좌측 상단의 x 좌표값
  - int nYOriginSrc : 좌측 상단의 y 좌표값
  - int nWidthSrc : 소스 사각형의 넓이
  - int nHeightSrc : 소스 사각형의 높이

  - UINT crTransparent : 투명하게 설정할 색상

투명 비트맵 처리

- 라이브러리 추가
  - **msimg32.lib 라이브러리**를 링크한다.
    - 프로젝트에 추가: 프로젝트 속성 → 링커 → 명령줄에서 라이브러리 추가
    - 또는 전처리기를 사용: #pragma comment (lib, "msimg32.lib")




```
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
       HDC hdc, memdc ;
        PAINTSTRUCT ps ;
        static HBITMAP hBitmap;

        switch (iMsg) {

        case WM_CREATE:
              hBitmap = (HBITMAP) LoadBitmap ( hInstance, MAKEINTRESOURCE(IDB_BITMAP7));
              break;

        case WM_PAINT:
              hdc = BeginPain t(hwnd, &ps);
              memdc=CreateCompatibleDC (hdc);
              SelectObject (memdc, hBitmap);

              TransparentBlt (hdc, 0, 0, 100, 100, memdc, 10, 50, 100, 100, RGB(0, 0, 0) );      //--- 마지막 인자를 RGB(0, 0, 0)으로 설정:  검정색을 투명하게 설정한다

              DeleteDC (memdc);
              EndPaint (hwnd, &ps);
              break;
       }
        return DefWinProc (hwnd, iMsg, wParam, lParam);
}
```

그 외 여러 비트맵 함수들

함수 설명	함수 프로토타입
점 찍기	COLORREF **SetPixel** (HDC hdc, int x, int y, COLORREF color);
점의 색상 알아보기	COLORREF **GetPixel** (HDC hdc, int x, int y);
지정한 사각 영역을 채색한다. (현재 DC에 선택되어 있는 브러시와 화면의 색상을 논리 연산)	BOOL **PatBlt** (HDC hdc, int x, int y, int nWidth, int nHeight, DWORD dwrop);
흑백 마스크 이미지를 사용하여 소스와 목적 색상을 혼합한다.	BOOL **MaskBlt** (HDC hdcDest, int xDest, int yDest, int width, int height, HDC hdcSrc, int xSrc, int ySrc, HBITMAP hbmMask, int xMask, int yMask, DWORD rop);
사변형 모양의 이미지 전송 (이미지 회전 가능)	BOOL **PlgBlt** (HDC hdcDest, const POINT *lpPoint, HDC hdcSrc, int xSrc, int ySrc, int width, int height, HBITMAP hbmMask, int xMask, int yMask);

반투명 픽셀 설정
 	
BLENDFUNCTION 구조체:
typedef struct BLENDFUNCTION {
BYTE BlendOp;
BYTE BlendFlags;
BYTE SourceConstantAlpha;
BYTE AlphaFormat;
};
	BOOL **AlphaBlend** (HDC hdcDest, int xoriginDest, int yoriginDest, int wDest, int hDest,
	HDC hdcSrc, int xoriginSrc, int yoriginSrc, int wSrc, int hSrc, BLENDFUNCTION ftn);
	
	//--- BLENDFUNCTION 설정 예) ➔ msimg32.lib 링크
	BLENDFUNCTION bf;
	
	bf.AlphaFormat = 0; // 일반 비트맵: 0, 32비트 비트맵: AC_SRC_ALPHA
	bf.BlendFlags = 0; // 무조건 0
	bf.BlendOp = AC_SRC_OVER; // 무조건 AC_SRC_OVER: 원본과 대상 이미지를 합침
	bf.SourceConstantAlpha = 127; // 투명도(투명 0 - 불투명 255)
	AlphaBlend (hDC, 0, 0, w, h, hMemDC, 0, 0, w, h, bf);

참고: CImage 클래스 사용하기

- **CImage는 그림 관련 클래스**
  - ATL에서 추가된 이미지 관리 클래스
    - ATL (Active Template Library): 작고 빠른 구성 요소 개체 모델 (COM, Component Object Model)을 만들 수 있도록 해주는 템플릿 기반의 C++ 클래스 집합
  - CImage는
    - 비트맵 관리 클래스, 비트맵 정보를 내부에서 보유
    - 여러 종류의 이미지 포맷을 지원: bmp 파일 외에 확장하여 png, jpg, gif 등의 다양한 포맷을 지원한다.
    - API 에서 지원되는 비트맵 관련 다양한 함수들이 지원된다: BitBlt, StretchBlt, TransparentBlt, AlphaBlend 같은 그림 관련 함수들이 지원된다.
  - atlImage.h 를 포함해야 한다.
  - 자세한 설명은
    - https://docs.microsoft.com/en-us/cpp/atl-mfc-shared/reference/cimage-class (영문)
    - https://msdn.microsoft.com/ko-kr/library/bwea7by5(v=vs.120).aspx (한글)

CImage 클래스 사용하기

- **CImage는 그림 관련 클래스**
  - 사용 가능한 public method들
    - Create (int nWidth, int nHeight, int nBitPerPixel, DWORD dwFlags);
      - Cimage 비트맵 생성
    - Destroy ();
      - Cimage 개체와 비트맵 삭제
    - Load (LPCTSTR pszFileName);
      - 이미지를 로드한다.
    - LoadFromResource (HINSTANCE hInstance, LPCTSTR pszResourceName);
      - 비트맵 리소스에서 이미지를 로드한다.
    - GetHeight (); GetWidth ();
      - 이미지 픽셀에서 높이/폭 값 리턴
    - Draw (HDC hDestDC, int xDest, int yDest, int nDestWidth, int nDestHeight, int xSrc, int ySrc, int nSrcWidth, int nSrcHeight);
      - 소스 사각형에서 대상 사각형으로 비트맵을 복사
    - BitBlt (HDC hDestDC, int xDest, int yDest, int nDestWidth, int nDestHeight, int xSrc, int ySrc, DWORD dwROP);
      - 소스 디바이스 컨텍스트에서 목적지 장치 컨텍스트로 비트맵을 복사
    - StretchBlt (HDC hDestDC, int xDest, int yDest, int nDestWidth, int nDestHeight, int xSrc, int ySrc, int nSrcWidth, int nSrcHeight, DWORD dwROP);
      - 소스 디바이스 컨텍스트에서 목적지 장치 컨텍스트로 비트맵을 크기 변경하여 복사
    - AlphaBlend (HDC hDestDC, int xDest, int yDest, int nDestWidth, int nDestHeight, int xSrc, int ySrc, int nSrcWidth, int nSrcHeight, BYTE bSrcAlpha = 0xff, BYTE bBlendOp = AC_SRC_OVER);
      - 투명하거나 반투명 이미지

CImage 클래스 사용하기

- **사용 예) Cimage를 사용하여 비트맵 출력하기**

```
#include <atlImage.h>

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam,  LPARAM lParam)
{
	PAINTSTRUCT ps;
	HDC hdc;
	CImage img;

	switch (message)
	{
		case WM_PAINT:
			hdc = BeginPaint (hWnd, &ps);
			img.Load (TEXT("bitmap1.png"));                    //--- bitmap1.png 파일을 로드하기
			nWidth = img.GetWidth();
			nHeight = img.GetHeight();

			img.Draw (hdc, 0, 0, rect.right, rect.bottom, 0, 0, nWidth, nHeight);
			//--- img.BitBlt (hdc, 0, 0, rect.right, rect.bottom, 0, 0, SRCCOPY);
			//--- img.StretchBlt (hdc, rect, SRCCOPY);

			EndPaint (hWnd, &ps);
			img.Destroy ();
		break;
	}
}
```

CImage 클래스 사용하기

- **사용 예) 더블 버퍼링 사용하기: 비트맵1을 배경으로 그리고 비트맵2가 튕기는 애니메이션 그리기**

```
#include <atlImage.h>

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam,  LPARAM lParam)
{
	static Cimage img, imgSprite;
	static int xPos=0, yPos=0;
	static RECT rect;
	switch (message)
	{
		case WM_CREATE:
			img.Load (L"bitmap1.bmp");                          //--- background
			imgSprite.Load (L"bitmap2.bmp");                    //--- sprite image
			GetClientRect (hWnd, &rect);
		break;
		case WM_PAINT:
			hdc = BeginPaint(hwnd, &ps);
			hBitmap = CreateCompatibleBitmap (hdc, rect.right, rect.bottom);
			memdc = CreateCompatibleDC (hdc);
			(HBITMAP) SelectObject (memdc, hBitmap);
			w = img.GetWidth();
			h = img.GetHeight();
			img.Draw (memdc, 0, 0, rect.right, rect.bottom, 0, 0, w, h);          //--- 메모리 DC에 배경 그리기
			imgSprite.Draw (memdc, xPos, yPos, 100, 100, 0, 0, 100, 100);         //--- 메모리 DC에 스프라이트 그리기
			BitBlt (hdc, 0, 0, rect.right, rect.bottom, memdc, 0, 0, SRCCOPY);    //--- 메모리 DC의 그림을 화면 DC에 복사하기
			DeleteObject (hBitmap);
			DeleteDC (memdc);
			EndPaint (hwnd, &ps);
		break;
		case WM_TIMER:
			xPos += 5;
			yPos += 5;
			InvalidateRect (hwnd, NULL, false);
		break;
	}
}
```

실습 5-4

- **이동하며 마우스에 반응하는 그림 그리기**
  - 배경 이미지를 출력한다.
  - 캐릭터 스프라이트가 애니메이션 되고, 좌우상하 중 한 방향으로 이동하고 있다. 가장자리에 도달하면 방향을 바꾼다.

  - **키보드 명령**
    - **←/→/↑/↓**: 스프라이트 캐릭터가 좌/우/상/하로 이동한다.
    - **j**: 스프라이트가 점프한다. 이동 방향의 수직 방향으로 점프한다. (점프할 때 다른 애니메이션 출력)
    - **e**: 이미지 크기가 확대됐다가 제자리로 돌아간다..
    - **s**: 이미지 크기가 줄어들었다가 제자리로 돌아간다.
    - **t**: 스프라이트 이미지를 복사 (트윈 이미지)하여 다른 곳에 애니메이션 한다. 위의 키보드 명령어를 입력하면 스프라이트 이미지 모두 명령어를 수행한다. 최대 5개의 트윈 이미지가 만들어진다.
    - **a**: 스프라이트 이미지가 빠른 속도로 지그재그 이동한다. 만들어진 트윈 이미지들은 메인 이미지의 경로를 따라 줄줄이 같이 이동한다./ 다시 누르면 멈추고 트윈 이미지들은 원래의 위치로 이동한다.
    - **r:** 지그재그 이동 방향을 유턴하며 반대 방향으로 바꾼다.

  - **마우스 명령**
    - **마우스를 이미지에 클릭하면,** 다른 애니메이션이 보여지고 캐릭터는 다른 곳으로 이동한다.
      - 이동된 위치에서 다시 원래의 캐릭터 애니메이션이 나타난다.
      - 원래 이미지에 적용되는 명령은 트윈 이미지에게도 똑같이 적용된다.
    - **마우스를 이미지가 없는 곳에 클릭하면,** 메인 캐릭터가 클릭한 곳으로 이동한다.
      - 이동 경로는 사선으로 진행된다 (가로, 세로로 이동하는 것이 아니라 직선으로 이동)

실습 5-5

- **움직이는 캐릭터 영역에 가두기**
  - 화면에 배경 그리기
  - 애니메이션 되는 캐릭터를 30개 화면의 위쪽에 그린다.
  - 사각형이 그려지면 떨어지는 캐릭터가 사각형 안에 들어오면 다른 애니메이션이 그려지고, 사각형 바닥에 멈춘다.
  - 사각형이 삭제되면 원래의 애니메이션이 그려지며 캐릭터들은 다시 아래로 떨어진다.
  - 캐릭터가 윈도우 아래에 도달하면 다시 위에서 떨어진다.
  - **왼쪽 마우스:**
    - 왼쪽 마우스를 드래그 하면 고무줄 효과가 있는 사각형이 그려진다.
  - **오른쪽 마우스를 누르고 드래그:**
    - 사각형이 선택되고 이동된다. 사각형이 이동되면 그 안의 캐릭터들의 위치도 같이 이동된다.
  - **키보드:**
    - **p:** 위에 있는 캐릭터들이 각각 다른 속도로 위에서 아래로 떨어진다. 이때 캐릭터는 애니메이션 되면서 떨어진다.
    - **d:** 그려진 사각형이 지워진다.
    - **r**: 리셋되고 다시 사각형을 그릴 수 있다.