

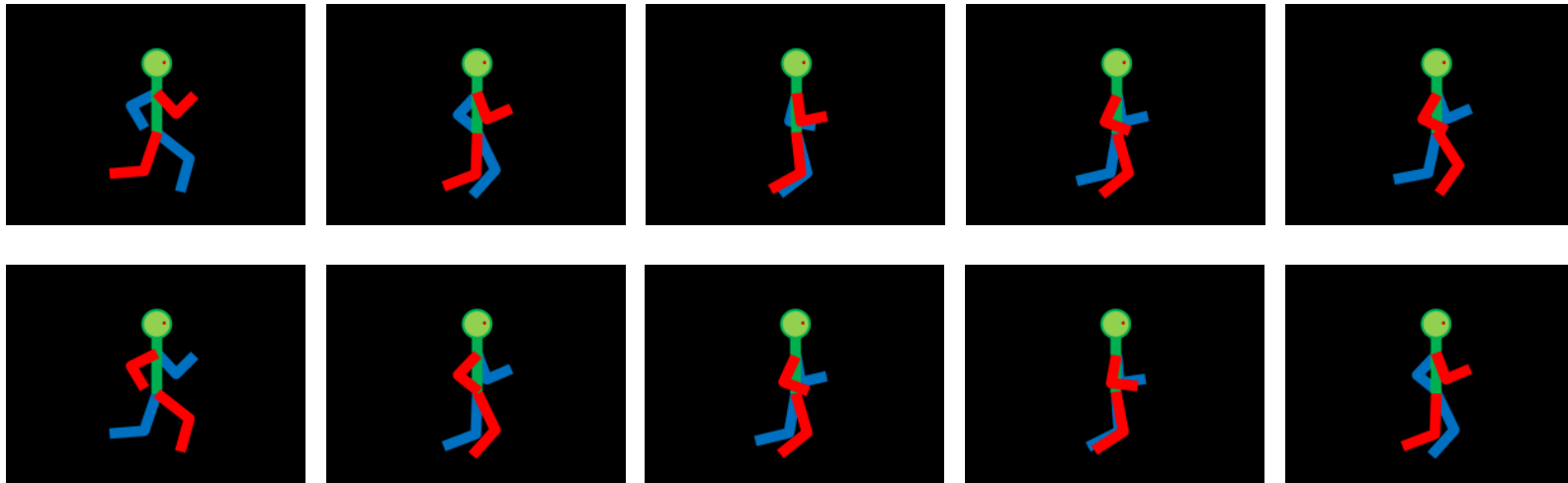
제 5장 비트맵과 애니메이션 2

2024년 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 **타이머**로 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 hBitCha[10];
```

```
int backWidth, backHeight, chaWidth, chaHeight;
```

```
int chaX, chaY, chaCount = 0;
```

```
LRESULT CALLBACK WndProc (HWND hWnd, UINT iMessage, WPARAM wParam, LPARAM lParam)
```

```
{
```

```
    HDC hDC;
```

```
    PAINTSTRUCT ps;
```

```
    BITMAP bmp;
```

```
    static int xPos = 0;
```

```
    switch (iMessage) {
```

```
    case WM_CREATE:
```

```
        hBitCha[0] = LoadBitmap(g_hInst, MAKEINTRESOURCE(IDB_BITMAP2));
```

```
        hBitCha[1] = LoadBitmap(g_hInst, MAKEINTRESOURCE(IDB_BITMAP3));
```

```
        hBitCha[2] = LoadBitmap(g_hInst, MAKEINTRESOURCE(IDB_BITMAP4));
```

```
        hBitCha[2] = LoadBitmap(g_hInst, MAKEINTRESOURCE(IDB_BITMAP5));
```

```
        ...
```

```
        hBitCha[9] = LoadBitmap(g_hInst, MAKEINTRESOURCE(IDB_BITMAP11));
```

```
        GetObject(hBitCha[0], sizeof(BITMAP), &bmp);
```

```
        chaWidth = bmp.bmWidth;
```

```
        chaHeight = bmp.bmHeight;
```

```
        SetTimer(hWnd, 1, 100, NULL);
```

```
    break;
```

//-- 필요한 10개의 애니메이션 이미지들을 로드하기

비트맵 애니메이션

//--- 10개의 이미지를 사용하여 애니메이션 구현

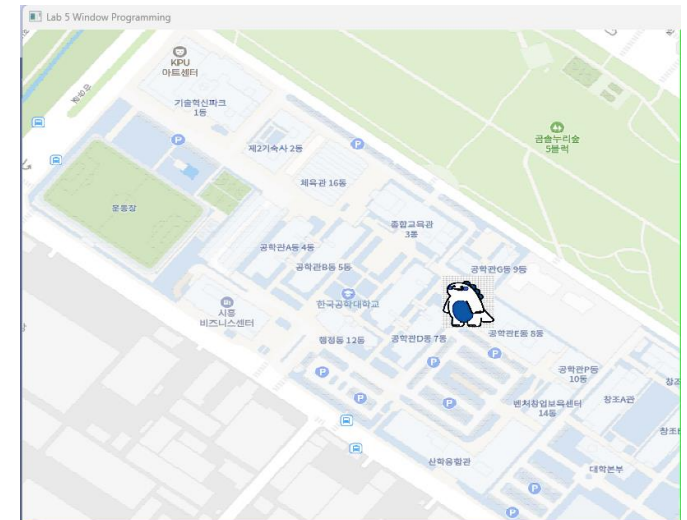
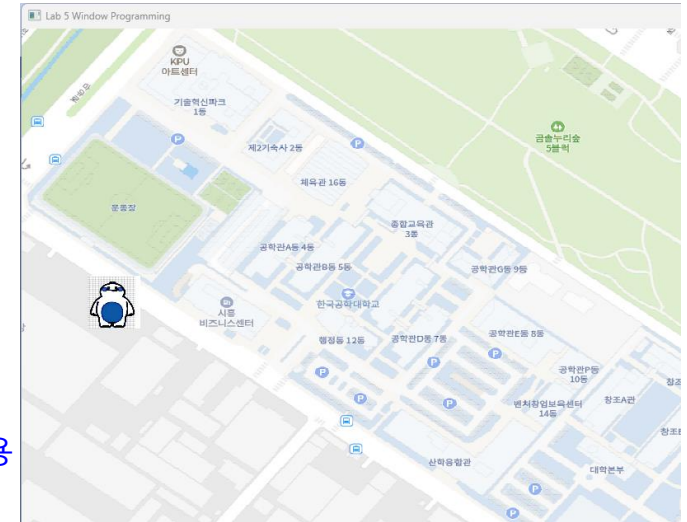
```
case WM_TIMER:  
    xPos += 10;  
    if (xPos > 800)  
        xPos = 0;  
    InvalidateRect(hWnd, NULL, true);  
return 0;
```

//--- 애니메이션의 x 위치 변경하기

```
case WM_PAINT:  
    hDC = BeginPaint(hWnd, &ps);  
    Animation(xPos, 300, hDC);  
    EndPaint(hWnd, &ps);  
break;
```

//--- (xPos, 300) 위치에 애니메이션 그리기: 사용자정의 함수 사용

```
case WM_DESTROY:  
    for (int i = 0; i < 10; i++)  
        DeleteObject(hBitCha[i]);  
    PostQuitMessage(0);  
}  
return DefWindowProc(hWnd, iMessage, wParam, lParam);  
}
```



비트맵 애니메이션

//--- 10개의 이미지를 사용하여 애니메이션 구현

```
void Animation (int xPos, int yPos, HDC hDC)
{
    HDC hmemDC;
    HBITMAP hBit, oldBit;
    static int count;
    int i;
    BITMAP bmp;

    count++;
    count = count % 10;

    hBit = LoadBitmap(g_hInst, MAKEINTRESOURCE (IDB_BITMAP1));           //--- 배경 이미지 로드하기
    GetObject(hBit, sizeof(BITMAP), &bmp);
    backWidth = bmp.bmWidth;
    backHeight = bmp.bmHeight;

    hmemDC = CreateCompatibleDC(hDC);

    oldBit = (HBITMAP)SelectObject(hmemDC, hBit);                         //--- 배경 이미지를 메모리 DC에 올리기
    StretchBlt(hDC, 0, 0, 800, 600, hmemDC, 0, 0, backWidth, backHeight, SRCCOPY);

    (HBITMAP)SelectObject(hmemDC, hBitCha[count]);                         //--- 순서대로 전경 이미지를 메모리 DC에 올리기
    StretchBlt(hDC, xPos, yPos, 64, 64, hmemDC, 0, 0, chaWidth, chaHeight, SRCCOPY);

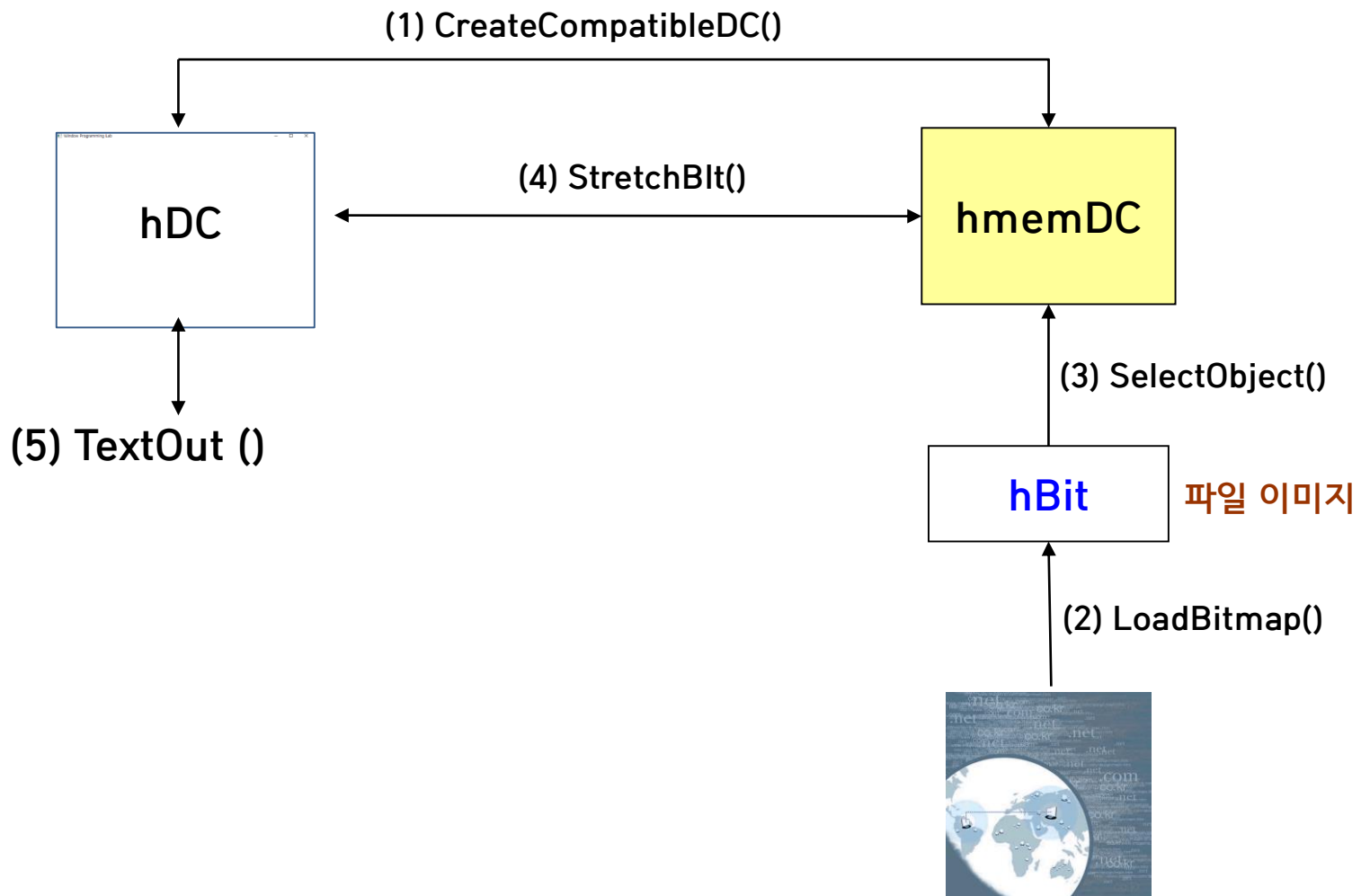
    SelectObject(hmemDC, oldBit);

    DeleteDC(hmemDC);                                                       //--- 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 **더블버퍼링**이라 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, hmemDC;  
    static HBITMAP hBit, oldBit;  
    TCHAR word[] = L"움직이는 그림";
```

```
    switch(iMsg) {
```

```
        case WM_CREATE:
```

```
            yPos = -30;                                     //--- -30: 글자의 높이 고려
```

```
            GetClientRect (hWnd, &rectView);
```

```
            hBit=LoadBitmap (g_hInst, MAKEINTRESOURCE(IDB_BITMAP1));
```

```
            SetTimer (hWnd, 1, 70, NULL);
```

```
            break;
```

```
        case WM_TIMER:
```

```
            //--- Timeout 마다 y좌표 변경 후, 출력 요청
```

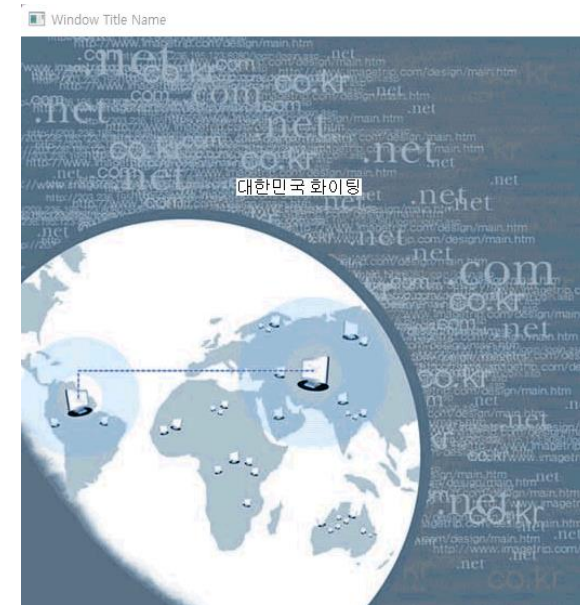
```
            yPos += 5;
```

```
            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 (g_hInst, MAKEINTRESOURCE(IDB_BITMAP1));
    hmemDC = CreateCompatibleDC (hDC);

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

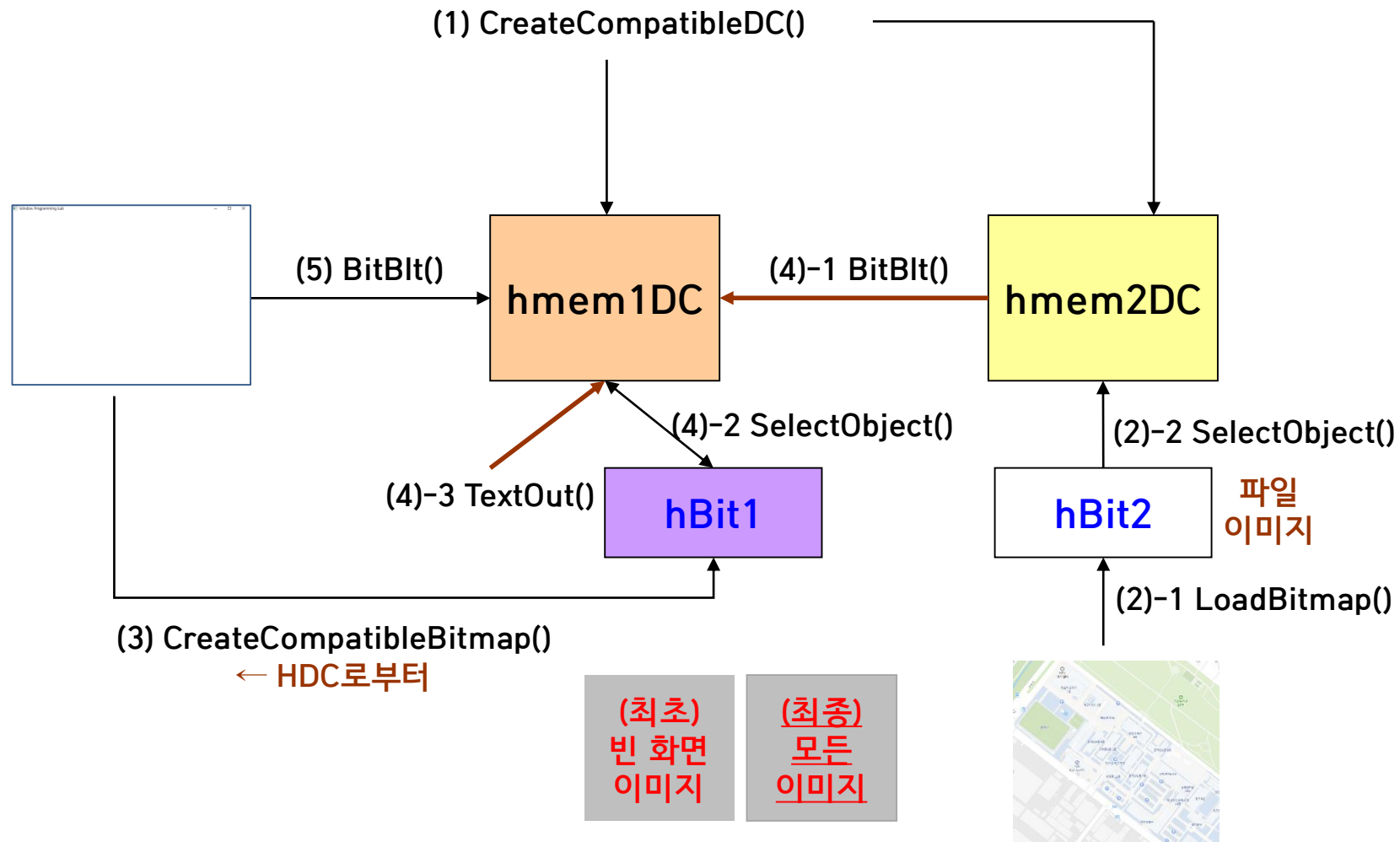
    //--- 메모리 DC → 화면 DC(hdc)로 이동, 출력
    StretchBlt (hDC, 0, 0, rectView.right, rectView.bottom, hmemDC, 0, 0, rectView.right, rectView.bottom, SRCCOPY);
    SelectObject (hmemDC, oldBit);
    DeleteDC (hmemDC);

    //--- 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, hmemDC;
```

```
    HDC hmem1DC, hmem2DC;
```

```
    PAINTSTRUCT ps;
```

```
    HBITMAP oldBit1, oldBit2;
```

```
    static RECT rectView;
```

```
    TCHAR word[] = L"Double Buffering";
```

```
    static int yPos;
```

```
    static HBITMAP hBit1, hBit2;
```

```
    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 ( g_hInst, MAKEINTRESOURCE(IDB_BITMAP1));
```

```
            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.

hmem1DC = CreateCompatibleDC (hDC);

//--- mem1dc와 호환되는 mem2dc를 만들어준다.

hmem2DC = CreateCompatibleDC (hmem1DC);

//--- mem2dc의 비트맵을 mem1dc에 옮기고, mem1dc를 hdc로 옮기려고 함

//--- hBit1의 이미지를 mem1dc로, hBit2의 이미지를 mem2dc로 선택

oldBit1 = (HBITMAP) SelectObject (hmem1DC, hBit1);

//--- hmem1DC 에는 hBit1

oldBit2 = (HBITMAP) SelectObject (hmem2DC, hBit2);

//--- hmem2DC 에는 hBit2: hBit2에는 배경 그림이 저장되어 있음

//--- mem2dc에 있는 배경그림을 mem1dc에 옮긴다.

BitBlt (hmem1DC, 0, 0, 1024, 768, hmem2DC, 0, 0, SRCCOPY);

SetBkMode (hmem1DC, TRANSPARENT);

TextOut (hmem1DC, 200, yPos, word, lstrlen(word));

//--- mem1dc에 텍스트 출력

//--- 저장한 비트맵 핸들값을 DC에 원상복귀, 생성된 MDC 삭제

SelectObject (hmem2DC, oldBit2);

DeleteDC (hmem2DC);

SelectObject (hmem1DC, oldBit1);

DeleteDC (hmem1DC);

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);
```

```
hmem1DC = CreateCompatibleDC (hDC);
```

//--- hBit1에는 배경과 텍스트가 출력된 비트맵이 저장되어 있다. 이 비트맵을 hmem1DC에 선택

```
oldBit1 = (HBITMAP) SelectObject (hmem1DC, hBit1);
```

//--- mem1dc에 있는 내용을 hdc에 복사한다.

```
BitBlt (hDC, 0, 0, 1024, 768, hmem1DC, 0, 0, SRCCOPY);
```

```
SelectObject (hmem1DC, oldBit1);
```

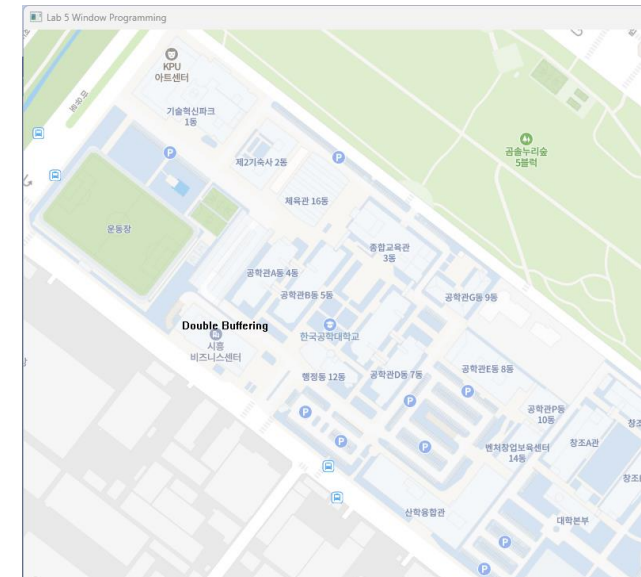
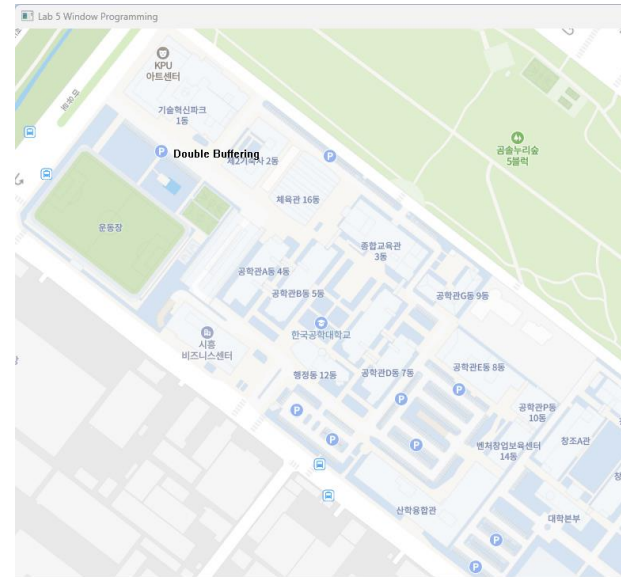
```
DeleteDC (hmem1DC);
```

```
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 DC를 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, hmemDC, hmemDCImage;

PAINTSTRUCT ps;

HBITMAP OldBit[3];

BITMAP bmp;

static HBITMAP hBitBackground, hBitCha[4];

static RECT rectView;

static HBITMAP hBit;

static int chaX, chaY, chaCount = 0;

static int backWidth, backHeight, chaWidth, chaHeight;

switch (iMessage) {

case WM_CREATE:

hDC = GetDC(hWnd);

GetClientRect (hWnd, &rectView);

hBitBackground = LoadBitmap (g_hInst, MAKEINTRESOURCE(IDB_BITMAP1));

//--- 배경 이미지 1개

hBitCha[0] = LoadBitmap (g_hInst, MAKEINTRESOURCE(IDB_BITMAP2));

//---- 애니메이션 이미지 3개

hBitCha[1] = LoadBitmap (g_hInst, MAKEINTRESOURCE(IDB_BITMAP3));

hBitCha[2] = LoadBitmap (g_hInst, MAKEINTRESOURCE(IDB_BITMAP4));

hBitCha[3] = LoadBitmap (g_hInst, MAKEINTRESOURCE(IDB_BITMAP5));

hBit = CreateCompatibleBitmap (hDC, rectView.right, rectView.bottom);

//--- 이미지들을 저장할 비트맵 생성: 전역변수 또는 static 타입으로 선언

GetObject (hBitBackground, sizeof(BITMAP), &bmp);

backWidth = bmp.bmWidth;

backHeight = bmp.bmHeight;

GetObject (hBitCha[0], sizeof(BITMAP), &bmp);

chaWidth = bmp.bmWidth;

chaHeight = bmp.bmHeight;

chaX = 0; chaY = 300;

SetTimer(hWnd, 1, 100, NULL);

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 > (rectView.bottom-5))                //--- static 변수로 선언한 애니메이션 x 좌표
        chaX = 0;

    //--- 더블 버퍼링 사용
    hmemDC = CreateCompatibleDC (hDC);              //--- 더블 버퍼로 사용 할 메모리 DC 생성
    hmemDCImage = CreateCompatibleDC (hmemDC);       //--- 이미지를 저장 할 메모리 DC 생성
    OldBit[0] = (HBITMAP)SelectObject (hmemDC, hBit); //--- MemDC와 hBit 를 호환되게 선택함

    OldBit[1] = (HBITMAP)SelectObject (hmemDCImage, hBitBackground); //--- 배경 이미지
    StretchBlt (hmemDC, 0, 0, windowX, windowY, hmemDCImage, 0, 0, backWidth, backHeight, SRCCOPY);

    OldBit[2] = (HBITMAP)SelectObject (hmemDCImage, hBitCha[chaCount]); //--- 애니메이션 이미지
    StretchBlt (hmemDC, chaX, chaY, 100, 100, hmemDCImage, 0, 0, chaWidth, chaHeight, SRCCOPY);
    //--- MemDC에 배경과 애니메이션 저장   hBit 비트맵에 저장

    chaCount++; //--- 애니메이션 이미지 순서
    chaCount %= 4;

    DeleteDC(hmemDC);
    DeleteObject(hmemDCImage);
    ReleaseDC(hWnd, hDC);

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


더블 버퍼링 적용 예 2) 배경화면 위로 움직이는 애니메이션

case WM_PAINT:

```
hDC = BeginPaint(hWnd, &ps);
```

```
//--- 더블 버퍼링 사용
```

```
hmemDC = CreateCompatibleDC (hDC);
```

```
OldBit[0] = (HBITMAP)SelectObject (hmemDC, hBit);
```

```
//--- hBit: 이미지가 저장되어 있는 비트맵 전역 변수
```

```
BitBlt(hDC, 0, 0, rectView.right, rectView.bottom, hmemDC, 0, 0, SRCCOPY);
```

```
SelectObject(hmemDC, OldBit[0]);
```

```
DeleteDC(hmemDC);
```

```
EndPaint(hWnd, &ps);
```

```
break;
```

case WM_DESTROY:

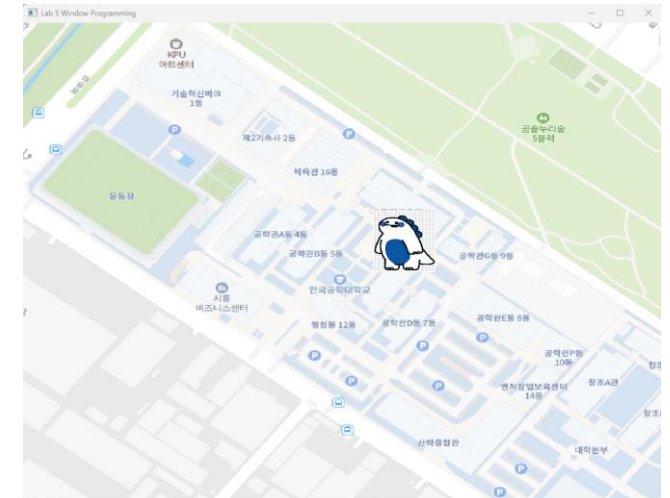
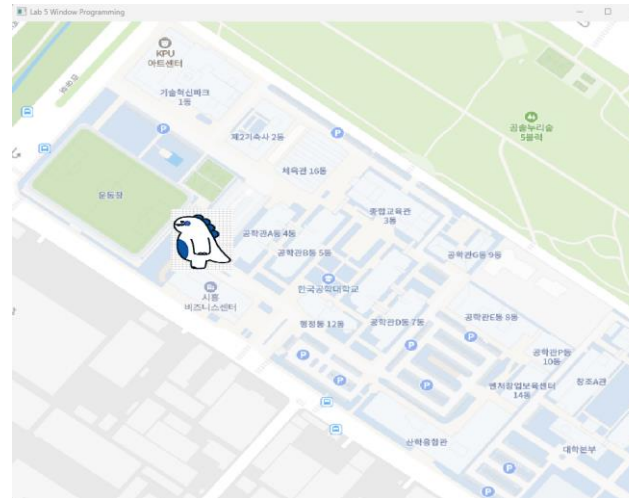
```
PostQuitMessage(0);
```

```
break;
```

```
}
```

```
return DefWindowProc(hWnd, iMessage, wParam, lParam);
```

```
}
```

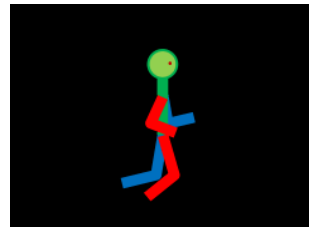
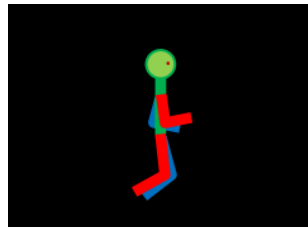
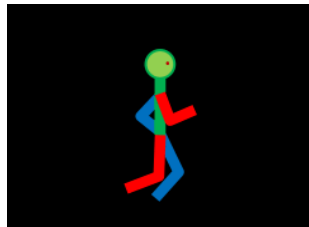


5. 비트맵 마스크

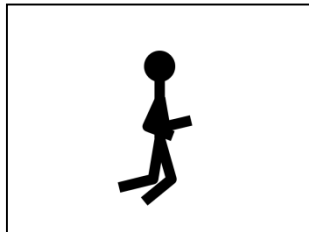
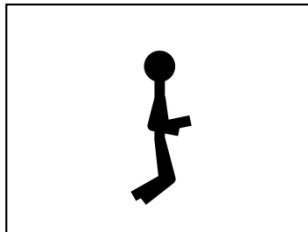
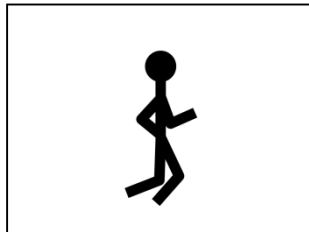
- 사각형의 비트맵 이미지에서 원하는 부분만을 사용하고 싶을 때, 그리려는 비트맵 이미지 부분에 마스크를 씌운다.

– 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.

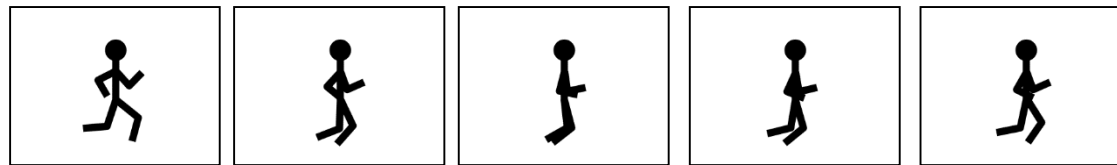
1. 소스의 원하는 부분을 흑백으로 처리한 마스크를 배경 그림과 **AND** 연산 → 배경 이미지에 흑색 마스크가 그려진다.

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

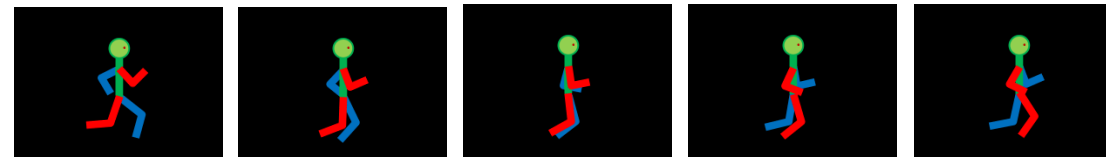
2. 여기에 원하는 그림을 배경 그림과 **OR** 연산 → 배경과 합성된 이미지로 나타나게 된다.

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

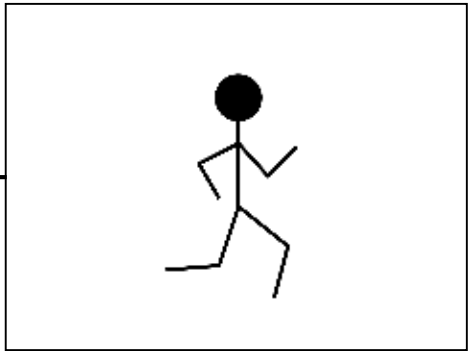
1. 마스크 그리기: AND 연산



컬러(X, X, X)

흰색(255,255,255)

AND



검은색(0, 0, 0)

	흰색(255,255,255)
AND	컬러(X , X , X)
<hr/>	
	컬러(X , X , X)

	검은색(0, 0, 0)
AND	컬러 (X, X, X)
<hr/>	
	검은색(0, 0, 0)

2. 캐릭터 그리기: OR 연산



컬러(X, X, X)
검은색(0, 0, 0)

OR



컬러(X, X, X)

	검은색(0, 0, 0)
OR	컬러 (X, X, X)
<hr/>	
	컬러 (X, X, X)

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=300;

    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, &ps);
        break;
    case WM_TIMER:
        xPos += 10;
        if ( xPos > windowWidth) xPos = 0;
        InvalidateRect (hWnd, NULL, true);
        break;
    case WM_DESTROY:
        for (i = 0; i < 5; 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  hmemDC;
```

```
    static int count;
```

```
    int   i;
```

```
    count++;
```

```
    count = count % 5;
```

```
    hmemDC = CreateCompatibleDC (hdc);
```

```
    hBit = LoadBitmap (hInst, MAKEINTRESOURCE(IDB_BITMAP5)); // 배경 이미지
```

```
    (HBITMAP) SelectObject (hmemDC, hBit);
```

```
    BitBlt (hdc, 0, 0, windowWidth, windowHeight, hmemDC, 0, 0, SRCCOPY);
```

마스크를 그린다.

```
    (HBITMAP) SelectObject (hmemDC, Mask[count]);
```

```
    BitBlt (hdc, xPos, yPos, 180, 240, hmemDC, 0, 0, SRCAND); //--- 180과 240은 이미지의 크기
```

```
    (HBITMAP) SelectObject (hmemDC, RunBit[count]);
```

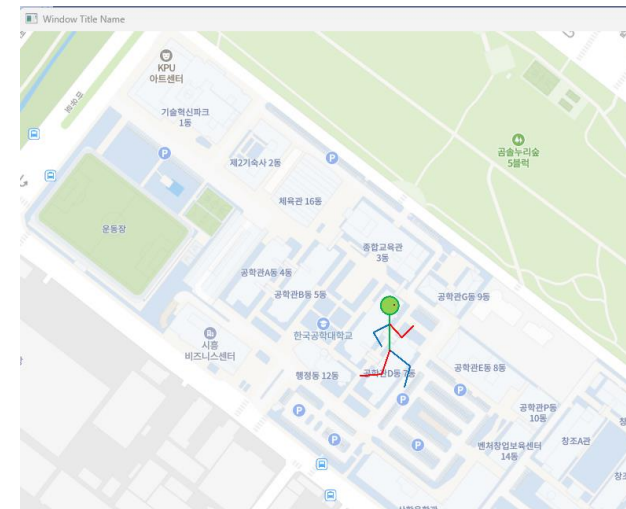
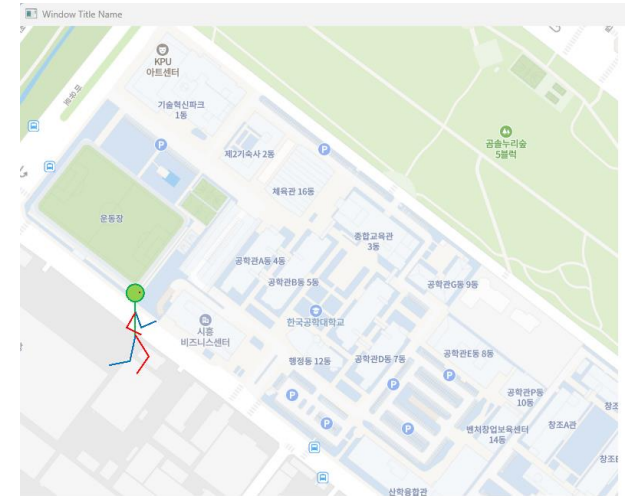
```
    BitBlt (hdc, xPos, yPos, 180, 240, hmemDC, 0, 0, SRCPAINT);
```

캐릭터를 그린다.

```
    DeleteDC (hmemDC);
```

```
    DeleteObject (hBit);
```

```
}
```



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 라이브러리를 링크한다.

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

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

```
{
```

```
HDC hDC, hmemDC ;  
PAINTSTRUCT ps ;  
static HBITMAP hBitmap;
```

```
switch (iMsg) {
```

```
case WM_CREATE:
```

```
hBitmap = (HBITMAP) LoadBitmap ( hInstance, MAKEINTRESOURCE(IDB_BITMAP7));  
break;
```

```
case WM_PAINT:
```

```
hDC = BeginPaint (hWnd, &ps);  
hmemdc=CreateCompatibleDC (hDC);  
SelectObject (hmemDC, hBitmap);
```

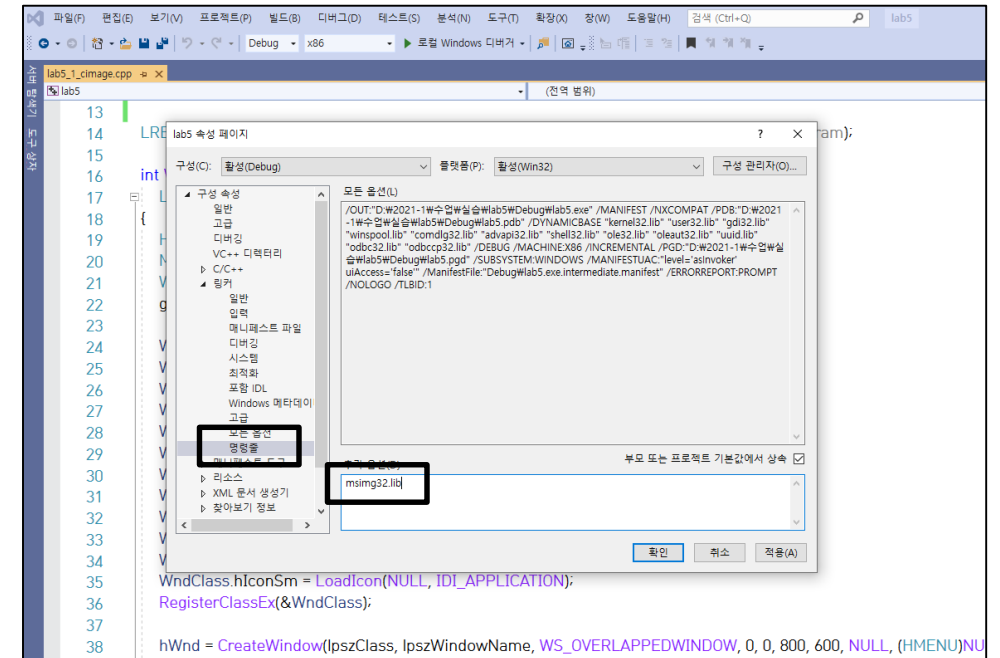
```
TransparentBlt (hDC, 0, 0, 100, 100, hmemDC, 10, 50, 100, 100, RGB(0, 0, 0) );
```

```
DeleteDC (hmemDC);  
EndPaint (hWnd, &ps);  
break;
```

```
}
```

```
return DefWinProc (hWnd, iMsg, wParam, lParam);
```

```
}
```



//--- 마지막 인자를 RGB(0, 0, 0)으로 설정: 검정색을 투명하게 설정한다

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; };	<pre>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);</pre>

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](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);
 - 비트맵 소스 사각형을 대상 사각형에 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 iMessage, WPARAM wParam, LPARAM lParam)
```

```
{  
    PAINTSTRUCT ps;  
    HDC hDC;  
    CImage img;  
    int nWidth, nHeight;  
    RECT rect;
```

```
    switch (iMessage)
```

```
    {  
        case WM_PAINT:  
            hDC = BeginPaint(hWnd, &ps);  
            img.Load (L"back.bmp");           //--- 이미지 파일을 로드하기  
            nWidth = img.GetWidth();           nHeight = img.GetHeight();  
            GetClientRect(hWnd, &rect);
```

```
            img.Draw (hDC, 0, 0, rect.right, rect.bottom, 0, 0, nWidth, nHeight);           //--- Draw 함수 사용하기  
            //--- img.BitBlt (hDC, 0, 0, rect.right, rect.bottom, 0, 0, SRCCOPY);           //--- BitBlt 함수 사용하기  
            //--- img.StretchBlt (hDC, 0, 0, rect.right, rect.bottom, 0, 0, nWidth, nHeight, SRCCOPY);           //--- StretchBlt 함수 사용하기
```

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

```
        case WM_DESTROY:  
            PostQuitMessage(0);  
            break;
```

```
    }  
    return DefWindowProc(hWnd, iMessage, wParam, lParam);  
}
```

CImage 클래스 사용하기

- 사용 예) 더블 버퍼링 사용하기: back.bmp를 배경으로 그리고 sprite.bmp가 튕기는 애니메이션 그리기

```
#include <atlImage.h>
```

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT iMessage, WPARAM wParam, LPARAM lParam)
```

```
{
```

```
    static CImage img, imgSprite;
```

```
    static int xPos = 0, yPos = 0;
```

```
    static RECT rect;
```

```
    HBITMAP hBitmap;
```

```
    HDC hDC, hmemDC;
```

```
    static int w, h, wSprite, hSprite;
```

```
    PAINTSTRUCT ps;
```

```
    static int xMove, yMove;
```

```
    switch (iMessage)
```

```
    {
```

```
        case WM_CREATE:
```

```
            img.Load (L"back.bmp");           //--- background
```

```
            imgSprite.Load (L"sprite.bmp"); //--- sprite image
```

```
            GetClientRect(hWnd, &rect);
```

```
            xMove = 5;
```

```
            yMove = 5;
```

```
            w = img.GetWidth();           //--- 배경 이미지 크기
```

```
            h = img.GetHeight();
```

```
            wSprite = imgSprite.GetWidth(); //--- 스프라이트 이미지 크기
```

```
            hSprite = imgSprite.GetHeight();
```

```
            SetTimer (hWnd, 1, 20, NULL);
```

```
            break;
```

CImage 클래스 사용하기

case WM_PAINT:

```
hDC = BeginPaint(hWnd, &ps);  
hBitmap = CreateCompatibleBitmap(hDC, rect.right, rect.bottom);  
hmemDC = CreateCompatibleDC(hDC);  
(HBITMAP)SelectObject(hmemDC, hBitmap);
```

```
img.Draw (hmemDC, 0, 0, rect.right, rect.bottom, 0, 0, w, h);  
imgSprite.Draw (hmemDC, xPos, yPos, wSprite, hSprite, 0, 0, wSprite, hSprite);  
BitBlt (hDC, 0, 0, rect.right, rect.bottom, hmemDC, 0, 0, SRCCOPY);
```

```
DeleteObject(hBitmap);  
DeleteDC(hmemDC);  
EndPaint(hWnd, &ps);  
break;
```

case WM_TIMER:

```
xPos += xMove;      yPos += yMove;  
if (xPos+wSprite > rect.right)    xMove *= -1;  //---x 위치  
Else if (xPos < 0)                xMove *= -1;  
If (yPos+hSprite > rect.bottom)   yMove *= -1;  //---y 위치  
else if (yPos < 0)                yMove *= -1;
```

```
InvalidateRect(hWnd, NULL, false);  
break;
```

case WM_DESTROY:

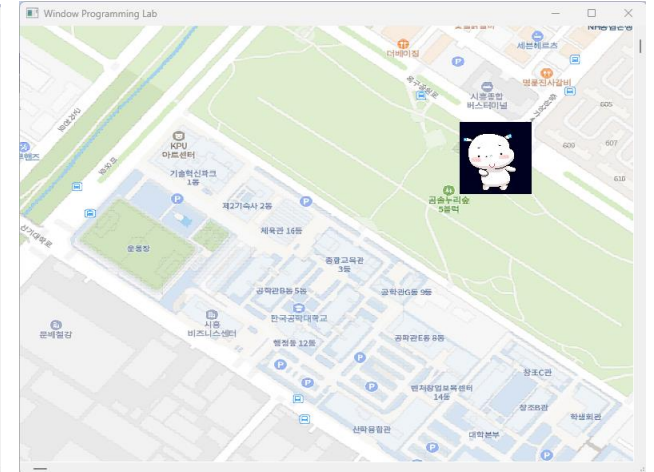
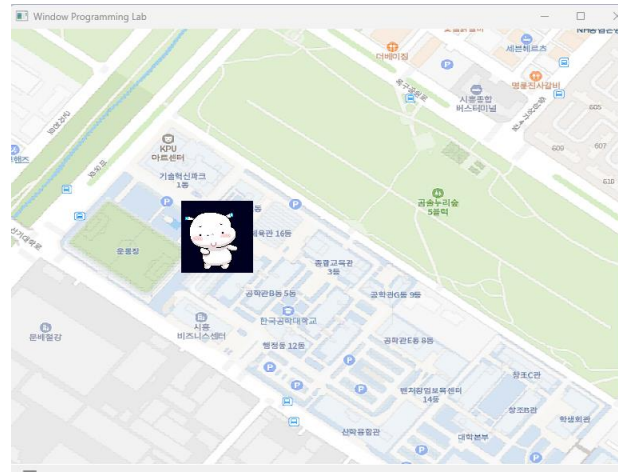
```
PostQuitMessage(0);  
break;
```

```
}
```

```
return DefWindowProc (hWnd, iMessage, wParam, lParam);
```

```
}
```

//--- 메모리 DC에 배경 그리기
//--- 메모리 DC에 스프라이트 그리기
//--- 메모리 DC의 그림을 화면 DC에 복사하기



실습 5-3

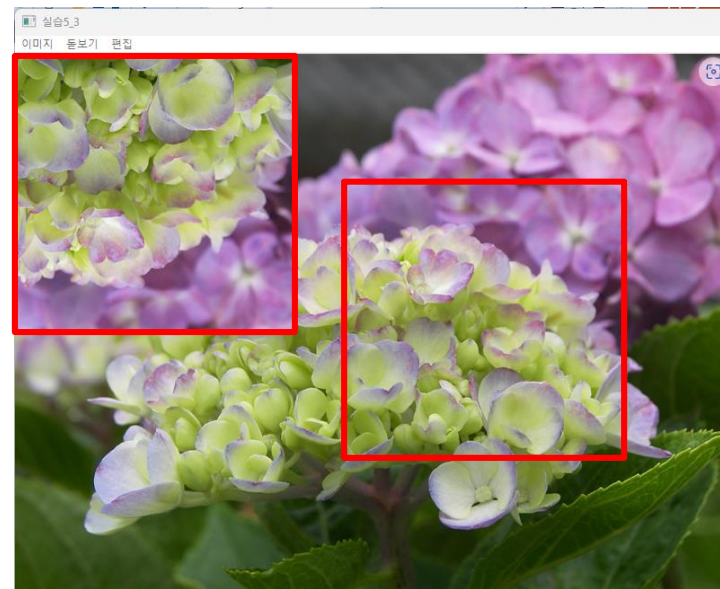
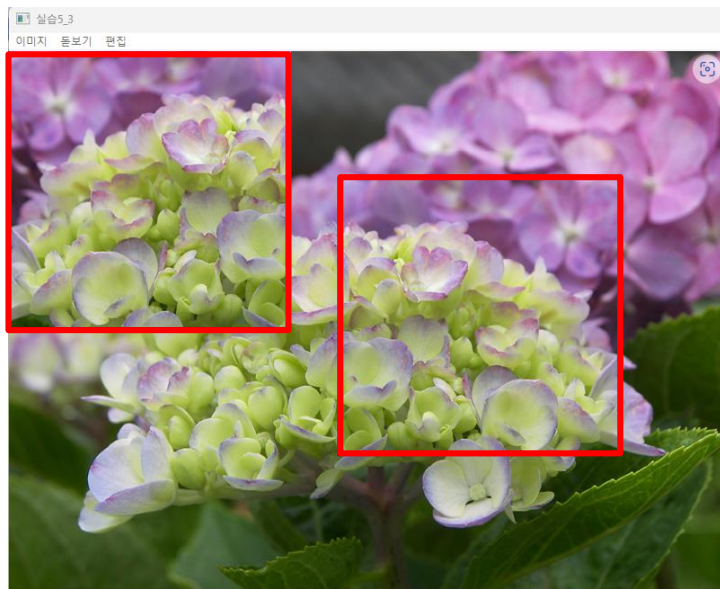
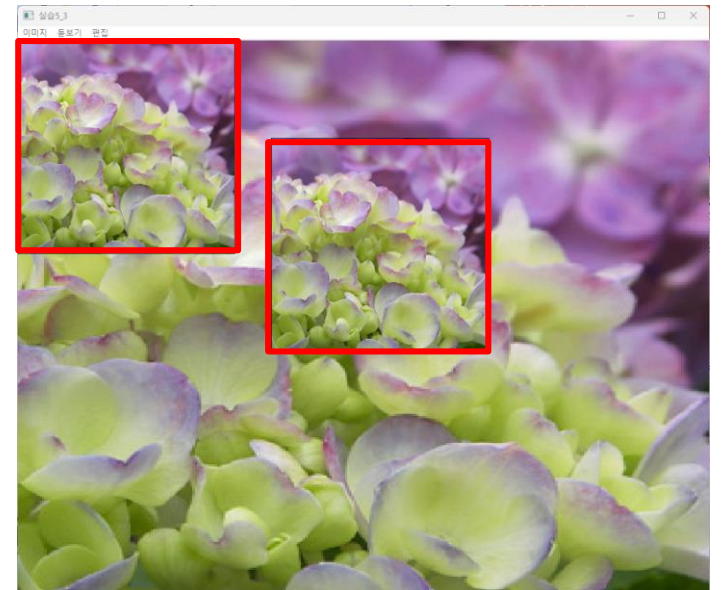
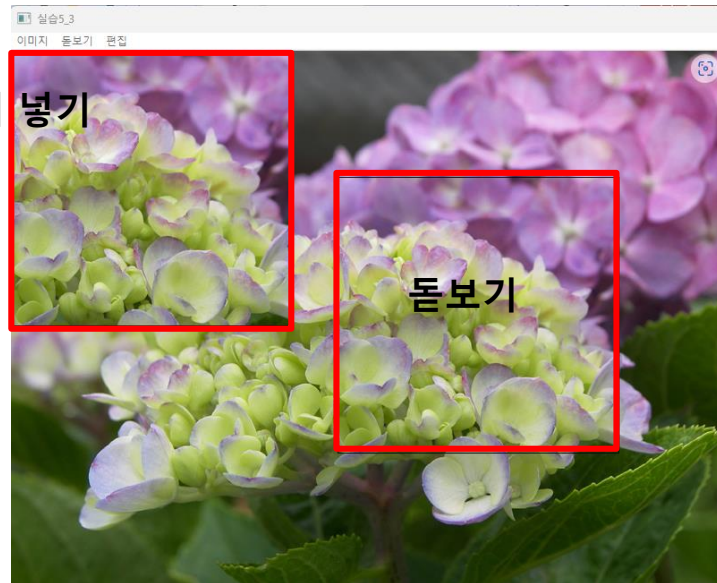
• 부분 이미지를 돋보기 보기

- 2개 이미지를 사용할 수 있게 한다.
- 화면에 디폴트로 설정된 이미지가 그려져 있다.
- **왼쪽 마우스를 클릭하고** 드래그하여 사각형(돋보기)을 그린다. (사각형은 고무줄 효과를 가진다)
- 기본 키보드 명령 (메뉴를 사용해도 됨)
 - **1/2**: 첫번째 이미지/두번째 이미지
 - **e**: 돋보기1- 돋보기 크기는 고정되고 고정된 돋보기 내에서 돋보기 내의 그림이 10%씩 확대해서 보인다.
 - **s**: 돋보기2- 돋보기 크기는 고정되고 고정된 돋보기 내에서 돋보기 내의 그림이 10%씩 축소해서 보인다.
 - **0**: 제자리- 돋보기 내의 그림이 원래의 크기로 보인다.
 - **c**: 복사- 돋보기 내의 그림이 복사된다.
 - **p**: 붙여 넣기- 돋보기 내의 그림이 (0, 0) 위치에 같은 크기로 복사된다.
 - **f**: 전체 붙여 넣기- 돋보기 내의 그림이 화면 전체에 복사된다. 다시 누르면 원래 그림으로 바뀐다.
 - **r**: 리셋 한다 (돋보기 사각형과 붙여 넣기 사각형이 없어지고 다시 돋보기 사각형을 입력 받게 한다.)
 - **q**: 프로그램 종료
- 그 외의 키보드 명령
 - **←/→/↑/↓**: 돋보기의 위치가 좌/우/상/하로 이동된다. 붙여 넣기 한 곳의 그림은 새로운 돋보기 위치의 그림으로 그려진다.
 - **m/n**: 돋보기 사각형이 작아진다/커진다.
 - **h/v**: 붙여 넣기 한 곳의 그림의 방향이 좌↔우 / 위↔아래로 반전된다.
 - **x/X**: 돋보기 사각형의 x축 크기가 축소/확대한다.
 - **y/Y**: 돋보기 사각형의 y축 크기가 축소/확대한다.
 - **a**: 돋보기 사각형이 지그재그 이동을 한다. 사각형 내의 그림이 사각형위치에 따라 바뀐다. (가로 우측 이동→ 한 칸 아래→ 가로 좌측 이동 → ...진행)
 - **b**: 돋보기 사각형이 커졌다 작아졌다 애니메이션 된다. 다시 누르면 애니메이션이 멈추고 원래 크기가 된다.
 - **i**: 돋보기 사각형 내의 그림이 반전되어 보인다/원래 색으로 보인다..
 - **k/K**: 붙여 넣기 한 곳의 그림을 점점 작게 그린다/점점 크게 그린다.
- 사각형(돋보기)를 **오른쪽 마우스로 클릭한 채로** 드래그하면 사각형의 위치가 바뀐다.

실습 5-3



붙여 넣기



실습 5-4

• 이동하며 마우스에 반응하는 캐릭터 애니메이션

- 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. 최대 5개의 트윈 이미지가 만들어진다.
- **0~5**: 원래 이미지와 스프라이트 이미지를 선택하고, 선택한 이미지를 표시하고 그 이미지에 대하여 점프, 확대, 축소 명령어를 적용한다.
- **a**: 스프라이트 이미지들이 모두 빠른 속도로 지그재그 이동한다. 만들어진 트윈 이미지들은 메인 이미지의 경로를 따라 줄줄이 같이 이동한다/다시 누르면 멈추고 트윈 이미지들은 원래의 위치로 이동한다.

- 마우스 명령

- **마우스를 이미지에 클릭하면**, 다른 애니메이션이 보여지고 캐릭터는 다른 곳으로 이동한다.
 - 이동된 위치에서 다시 원래의 캐릭터 애니메이션이 나타난다. 메인 이미지가 이동하면 트윈들은 따라 이동한다.
 - 트윈 이미지를 클릭하면, 트윈 이미지 애니메이션 대신 다른 애니메이션은 보여지지만 이동하지는 않는다.
- **마우스를 이미지가 없는 곳에 클릭하면**, 메인 캐릭터가 클릭한 곳으로 이동한다.
 - 이동 경로는 사선으로 진행된다 (가로, 세로로 이동하는 것이 아니라 직선으로 이동)